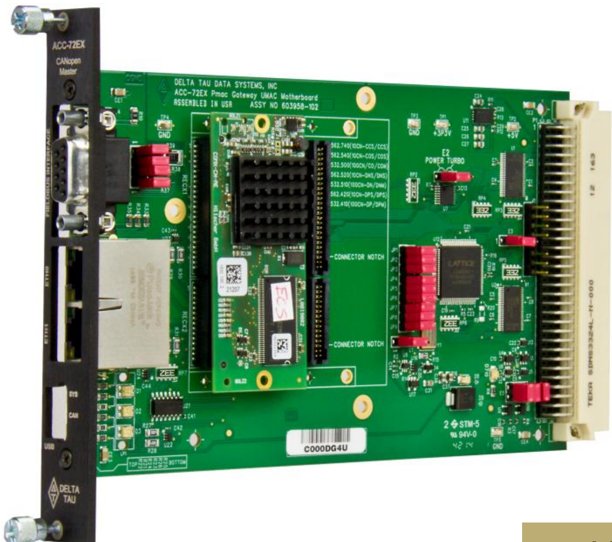
USER MANUAL

Accessory 72EX



UMAC Fieldbus Interface

3x0-603958-xUxx

September 1, 2015

DELTA TAU
Data Systems, Inc.
NEW IDEAS IN MOTION ...

Single Source Machine Control *Power // Flexibility // Ease of Use*
21314 Lassen Street Chatsworth, CA 91311 // Tel. (818) 998-2095 Fax. (818) 998-7807 // www.deltatau.com

Copyright Information

To report errors or inconsistencies, call or email:

Delta Tau Data Systems, Inc. Technical Support
Phone: (818) 717-5656
Fax: (818) 998-7807
Email: support@deltatau.com
Website: http://www.deltatau.com

Operating Conditions

All Delta Tau Data Systems, Inc. motion controller products, accessories, and amplifiers contain static sensitive components that can be damaged by incorrect handling. When installing or handling Delta Tau Data Systems, Inc. products, avoid contact with highly insulated materials. Only qualified personnel should be allowed to handle this equipment.

In the case of industrial applications, we expect our products to be protected from hazardous or conductive materials and/or environments that could cause harm to the controller by damaging components or causing electrical shorts. When our products are used in an industrial environment, install them into an industrial electrical cabinet or industrial PC to protect them from excessive or corrosive moisture, abnormal ambient temperatures, and conductive materials. If Delta Tau Data Systems, Inc. products are directly exposed to hazardous or conductive materials and/or environments, we cannot guarantee their operation.

Warning

A Warning identifies hazards that could result in personal injury or death. It precedes the discussion of interest.

Caution

A Caution identifies hazards that could result in equipment damage. It precedes the discussion of interest.

Note

A Note identifies information critical to the understanding or use of the equipment. It follows the discussion of interest.

REV	DESCRIPTION	DATE	CHG	APPVD
1	Preliminary Manual	11/05/12	Sina Sattari	SS
2	Added Power PMAC support and address settings based upon 603958-102	09/24/13	Sina Sattari	SS
3	Corrected **ACC72EX.Data8[*i*]** references	10/21/2013	Sina Sattari	SS
4	Added C code and setup examples; corrected typos	7/29/15	DCDP	SS

Table of Contents

INTRODUCTION

This manual provides the information needed to configure ACC-72EX, a fieldbus/real-time Ethernet interface for the Turbo or Power UMAC. The ACC-72EX is equipped with a "gateway" daughter card that allows the UMAC (also referred to as host application) to send and receive data through the supported fieldbus/real-time Ethernet protocols. The gateway used is the COMX CN series manufactured by the Hilscher Corporation. Relevant hyperlinks are provided in Appendix D for in-depth information regarding these modules.

There are three connectors located on the front of the ACC-72EX:

First, a Micro B USB connector, which is specified as "Diagnostic Port," and provides USB connectivity to Hilscher's "SyCon.NET" software.

The second connector, which is referred to as the "Fieldbus Port", is a 9-Pin Male D-Sub connector which is used for connecting the fieldbus link to ACC-72EX. The fieldbus protocols supported through this port are:

- PROFIBUS-DP – Master – OPT10
- PROFIBUS-DP – Slave – OPT11
- DeviceNet – Master – OPT20
- DeviceNet – Slave – OPT21
- CANopen – Master – OPT30
- CANopen – Slave – OPT31
- CC-Link – Slave – OPT51

The third connector is composed of two RJ-45 ports which provide connection to real-time Ethernet networks. The following real-time Ethernet protocols are supported through these ports:

- EtherCAT – Master – OPT60
- EtherCAT – Slave – OPT61
- EtherNet/IP – Scanner/Master – OPT70
- EtherNet/IP – Adaptor/Slave – OPT71
- Open Modbus/TCP – OPT80
- PROFINET IO – Controller – OPT90
- PROFINET IO – Device – OPT91

The protocol is dependent upon the equipped COMX gateway. The hardware cannot be programmed for an alternate protocol or change from slave to master or vice versa. However, should the COMX gateway be replaced with one supporting another protocol, the baseboard would function properly as a communications link to UMAC. In this case, proper jumper settings should be set up to ensure proper functionality on communication lines and option detection.

Most gateway cards get their power from the UBUS back plane; however, the DeviceNet option (Options 3 & 4) requires an external 24 VDC power supply through the "Fieldbus Port."

THEORY OF OPERATION

The ACC-72EX board is organized as a motherboard/daughter board system. The motherboard contains the UBUS interface, diagnostics, and the fieldbus connections. The daughter board contains the intelligence (firmware which will be referred as netX) and the interface electronics required for each fieldbus. There is a different daughter board for each fieldbus.

The netX firmware on the daughter board implements each fieldbus communications protocol. Fieldbus data is transferred to/from the fieldbus and placed in a Dual-Ported RAM (memory) on the daughter board. The structure of this DPRAM is given later in this manual and is common for all the field buses. ACC-72EX supports up to 64K DPRAM on each device (one full chip-select width).

The PMAC side of the DPRAM is interfaced to the UBUS. PMAC programs access the fieldbus data by reading or writing data to memory addresses corresponding to the location of the PMAC Gateway 3U board's DPRAM.

UBUS Interface

The UBUS is Delta Tau's bus interface for the UMAC controller. The ACC-72EX maps to the UBUS as a DPRAM style board. It occupies contiguous memory locations (both X and Y memory for Turbo PMAC) of the lower two bytes of the 24-bit (middle 16 bits of each 32 bit word for Power PMAC), DPRAM addresses. Because the DPRAM size supported on ACC-72EX can be as large as 64K, each card will occupy one full Chip Select addressing space. There can be a maximum of two ACC-72EX cards per Turbo/Power UMAC (cannot be in a MACRO Station).

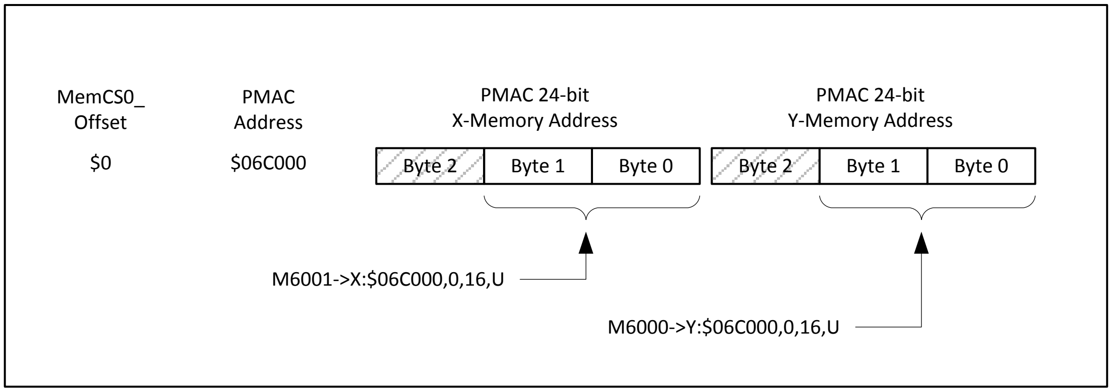
M-Variables can be mapped to these locations to move data to and from the fieldbus and PMAC. In addition to fieldbus data, there is a block of memory that indicates the ACC-72EX's status.

How ACC-72EX Works

1. The ACC-72EX organizes fieldbus bytes in dual-port memory on the COMX module. These fieldbus bytes are mapped into PMAC's memory space via the UBUS interface.
2. PMAC M-Variables are used to move data to and from the fieldbus or to control the COMX board.
3. An E-point jumper on the ACC-72EX sets the address of the board in PMAC memory space.
4. The COMX board is configurable via a USB port. SYCON.NET is provided with the COMX board for this purpose.
5. Diagnostic LEDs are provided for a visual indication of board status.

Turbo PMAC Memory

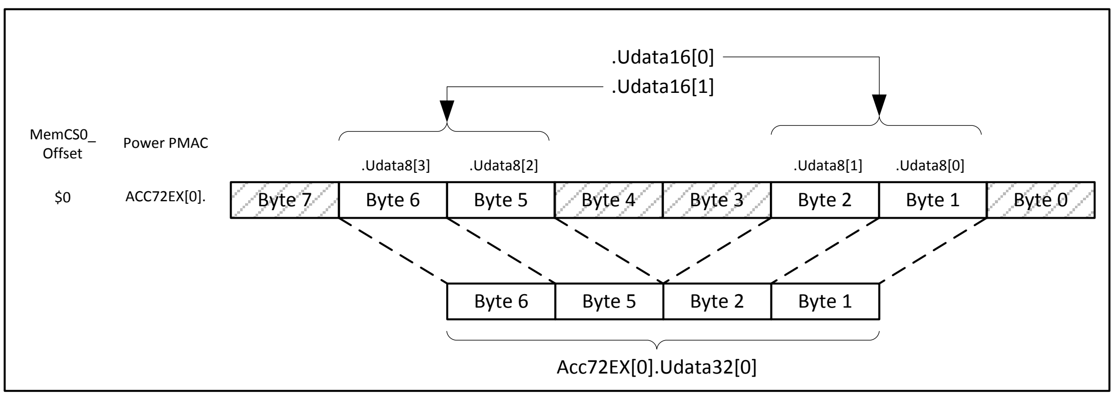
Turbo PMAC uses a DSP (Digital Signal Processor) with a 24-Bit architecture that uses two memory areas: Y and X Memory. Memory is accessed in PMAC programs using M-Variables. The definition of an M-Variable includes its number, address, offset, width, and type. Refer to the Turbo PMAC Software Reference Manual or Turbo PMAC User Manual for additional explanation of M-Variables and their specification, such as in the "M-Variables" section in the User manual.



Turbo PMAC Memory Organization

Power PMAC Memory

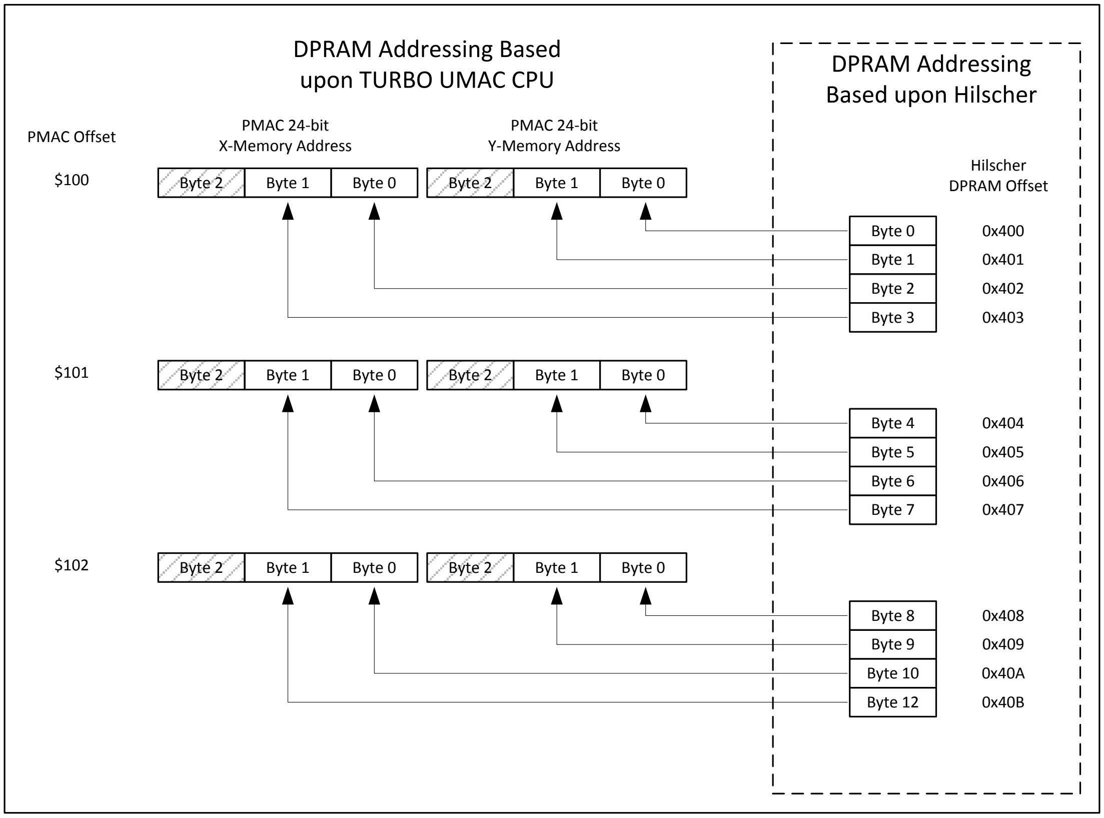
Power PMAC uses 32-bit data bus architecture. ACC-72EX Memory is accessed in Power PMAC data structures or their equivalent **#define** statements. The **#define** statements are included later in this manual.



Power PMAC Memory Organization

Hilscher ComX Module Addressing to Turbo PMAC Addressing Conversion

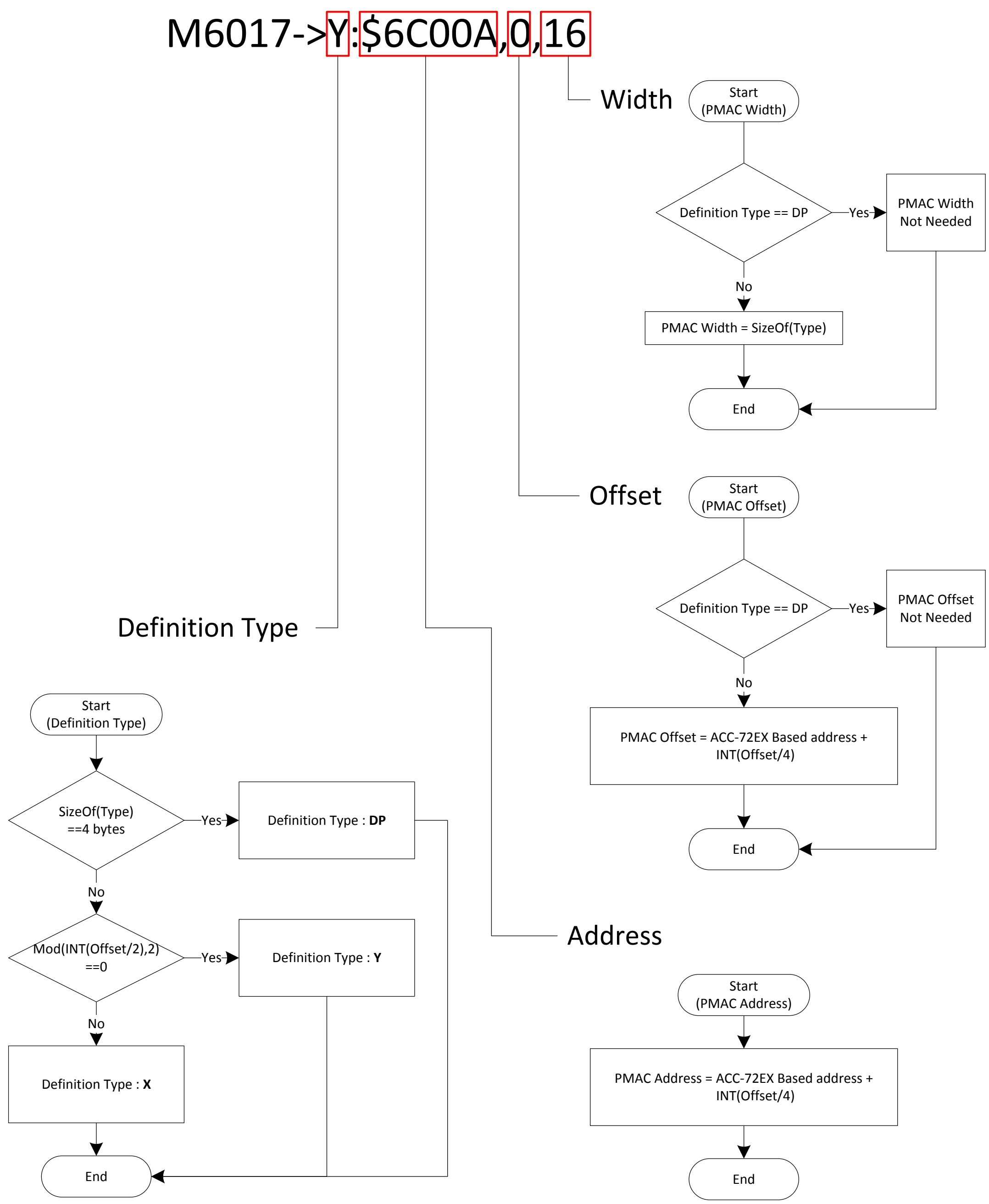
As explained in previous sections, Turbo PMAC places 4 bytes of Hilscher ComX memory data in each PMAC memory word. This means that for each address offset increment on the PMAC side, there will be 4 increments of offset addresses on the Hilscher DPRAM side. The following example shows PMAC addressing for equivalent offset addresses of 0x400 in Hilscher documentation.



Consumed Data Flow

In general, the following flowcharts can be used to convert any Hilscher DPRAM addressing to PMAC's addressing format:

System Information Block			
Offset	**Type**	**Name**	**Description**
0x0028	UINT16	`usDeviceClass`	Device Class netX Device Class (see page 34)

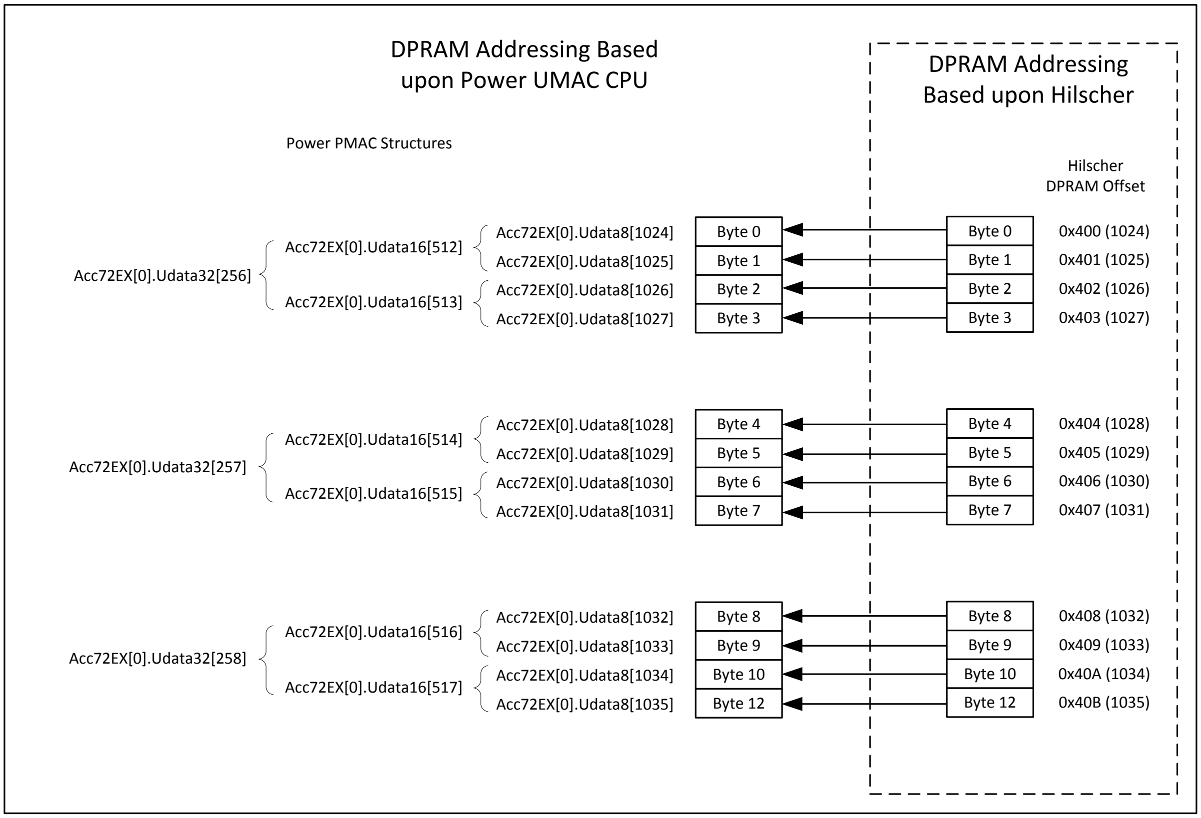


An address conversion tool is provided in the ACC-72EX Setup Assistant software.

Hilscher ComX Module Addressing to Power PMAC Addressing Conversion

In Power PMAC, specific **Acc72EX[*i*]** data structures have been implemented which allow bit-wide, byte-wide, 2-byte and 4-byte access to Hilscher ComX Dual Ported RAM.

Acc72EX[*i*].Udata16[*j*] structures can be used for individual bit access, both for read and for write purpose.



HARDWARE

E3: UBUS Address

E-point jumper E3 on the ACC-72EX controls the base address and range on the UBUS. Since each ACC-72EX uses full-13 bit addressing, it consumes all the memory addressable through each chip select. As a result, two is the maximum number of ACC-72EX boards that can be used in a Turbo UMAC rack.

E3	Turbo PMAC	Power PMAC
1-2	Y/X:$6C000 - $6FFFF	ACC-72EX[0] ($E00000)
2-3	Y/X:$74000 - $7FFFF	ACC-72EX[1] ($F00000)

The default location on Turbo PMAC is Y/X:$6C000 - $6FFFF ($E00000 on Power PMAC).

Note:

Do not set the ACC-72EX to the DPR address range $6C000-$6FFFF if the UMAC is equipped with an Acc-54E. Acc-54E is set to this range as default.

CS16- Identification

One of the features of the UBUS is that memory locations, selected by CS16 (Chip Select 16/Active Low), were reserved for board identification information.

- Vendor ID (8 bits)
- Options Present (10 bits)
- Revision Number (4 bits)
- Product ID (14 bits)

This information (36 bits) is accessible directly with I-Variables added in Turbo PMAC Firmware 1.936 or later. A summary of the PMAC Gateway ID information is in the table below.

I39 controls the values reported.

I39=	I4942…I4952 reports the following
0	36 bits (Vendor ID, Options present, Rev Number, Product ID)
1	8 bits (Vendor ID)
2	10 bits (Options Present) Reported by PMAC in HEX ($)
3	4 bits (Revision Number)
4	14 bits (Product ID)
5	19 bits (Card Base Address)

Identification Information

The vendor ID, part number, and revision numbers are programmed into the ACC-72EX base board. The Option Number is set by jumpers on the board. The settings below are given for reference only. There is no need to change these from the factory settings.

Jumper Settings

Option Identification Jumpers

Item	Comm. Protocol Option	Part Number	JP1	JP2	JP3	JP4	JP5	JP6	JP7	JP8	JP9
			Bit 9	Bit 10	Bit 11	Bit 12	Bit 13	Bit 14	Bit 15	Bit 16	Bit 17
1	PROFIBUS-DP – Master	310-603958-OPT	OFF	**ON**	OFF	**ON**	OFF	OFF	OFF	OFF	OFF
2	PROFIBUS-DP – Slave	311-603958-OPT	**ON**	**ON**	OFF	**ON**	OFF	OFF	OFF	OFF	OFF
3	DeviceNet – Master	320-603958-OPT	OFF	OFF	**ON**	OFF	**ON**	OFF	OFF	OFF	OFF
4	DeviceNet – Slave	321-603958-OPT	**ON**	OFF	**ON**	OFF	**ON**	OFF	OFF	OFF	OFF
5	CANopen – Master	330-603958-OPT	OFF	**ON**	**ON**	**ON**	**ON**	OFF	OFF	OFF	OFF
6	CANopen – Slave	331-603958-OPT	**ON**	**ON**	**ON**	**ON**	**ON**	OFF	OFF	OFF	OFF
7	CC-Link – Slave	351-603958-OPT	**ON**	**ON**	OFF	OFF	**ON**	**ON**	OFF	OFF	OFF
8	EtherCAT – Master	360-603958-OPT	OFF	OFF	**ON**	**ON**	**ON**	**ON**	OFF	OFF	OFF
9	EtherCAT – Slave	361-603958-OPT	**ON**	OFF	**ON**	**ON**	**ON**	**ON**	OFF	OFF	OFF
10	EtherNet/IP – Scanner/Master	370-603958-OPT	OFF	**ON**	**ON**	OFF	OFF	OFF	**ON**	OFF	OFF
11	EtherNet/IP – Adaptor/Slave	371-603958-OPT	**ON**	**ON**	**ON**	OFF	OFF	OFF	**ON**	OFF	OFF
12	Open Modbus/TCP	380-603958-OPT	OFF	OFF	OFF	OFF	**ON**	OFF	**ON**	OFF	OFF
13	PROFINET IO – Controller	390-603958-OPT	OFF	**ON**	OFF	**ON**	**ON**	OFF	**ON**	OFF	OFF
14	PROFINET IO – Device	391-603958-OPT	**ON**	**ON**	OFF	**ON**	**ON**	OFF	**ON**	OFF	OFF

E-Point Jumper Settings

Point	Default	Description
E1	1-2	Selection of Reset Polarity Signal for Hilscher Module: 1-2 Selects Low True Reset 2-3 Selects High True Reset
E2	2-3	Selection of UMAC CPU architecture. This selection affects the data bus and provides contiguous data addressing for the DPRAM: 1-2 Power PMAC 2-3 Turbo PMAC
E3	1-2	Selection of base address for ACC-72EX: 1-2 Selects Y/X:$6C000 - $6FFFF (**Acc72EX[0]**) 2-3 Selects Y/X:$74000 - $7FFFF (**Acc72EX[1]**)
E5	OFF	Connects DPRAM interrupt to UBUS IRQ-1
E6	OFF	Connects DPRAM interrupt to UBUS IRQ-2
E7	OFF	Connects DPRAM interrupt to UBUS IRQ-3

Communication Option-Dependent E-Point Jumper Settings

Comm. Protocol Option	Option Part Number	E8	E10	E11	E12
PROFIBUS-DP – Master	310-603958-OPT	1-2	OFF	OFF	OFF
PROFIBUS-DP – Slave	311-603958-OPT	1-2	OFF	OFF	OFF
DeviceNet – Master	320-603958-OPT	2-3	ON	OFF	OFF
DeviceNet – Slave	321-603958-OPT	2-3	ON	OFF	OFF
CANopen – Master	330-603958-OPT	OFF	OFF	ON	OFF
CANopen – Slave	331-603958-OPT	OFF	OFF	ON	OFF
CC-Link – Slave	351-603958-OPT	2-3	OFF	OFF	ON
EtherCAT – Master	360-603958-OPT	OFF	OFF	OFF	OFF
EtherCAT – Slave	361-603958-OPT	OFF	OFF	OFF	OFF
EtherNet/IP – Scanner/Master	370-603958-OPT	OFF	OFF	OFF	OFF
EtherNet/IP – Adaptor/Slave	371-603958-OPT	OFF	OFF	OFF	OFF
Open Modbus/TCP	380-603958-OPT	OFF	OFF	OFF	OFF
PROFINET IO – Controller	390-603958-OPT	OFF	OFF	OFF	OFF
PROFINET IO – Device	391-603958-OPT	OFF	OFF	OFF	OFF

NOTES:

E8: Determines the signal on pin 5 of the Fieldbus 9-pin D-Sub Connector. The position of the jumper depends on the COMX module installed/option ordered.
E10: Adds 120 Ω termination resistor for DeviceNet communication lines
E11: Adds 120 Ω termination resistor for CANopen communication lines
E12: Adds 110 Ω termination resistor for CC-Link communication lines

Connector Pinouts

Fieldbus Port (J4)

Protocol / Pin No.	PROFIBUS	DeviceNet	CANopen	CC-Link
1		+24 V Power Supply		CC-Link, Shield
2	Positive power supply	CAN High-Signal	CAN_L Bus Line	CC-Link, Function Ground
3	Receive / Send Data-P	Reference potential	CAN Ground	CC-Link, Data A
4	Control			
5	Reference potential	Shield		CC-Link, Data Ground
6	Positive power supply	CAN High-Signal	CAN_L Bus Line	CC-Link, Function Ground
7			CAN_H Bus Line	
8	Receive / Send Data-N			
9		CAN Low-Signal		CC-Link, Data B
NOTES	E8, Jumpered 1-2	E8, 2-3 Jumpered E10 Jumpered	E11 Jumpered	E8, Jumpered 2-3 E12 Jumpred

Real-time Ethernet Ports (Ethernet 0 & Ethernet 1)

Pin No.	Symbol	Description
1	RX+	Receive+
2	RX-	Receive–
3	TX+	Transmit+
4		
5		
6	TX-	Transmit–
7		
8		

Diagnostics Port (Micro A USB)

Pin No.	Symbol	Description
1	VBUS	+5 VDC (Not connected to ACC-72EX +5 VDC)
2	D-	Data -
3	D+	Data +
4	GND	Ground Reference (Connected to ACC-72EX and UMAC's Digital Ground)

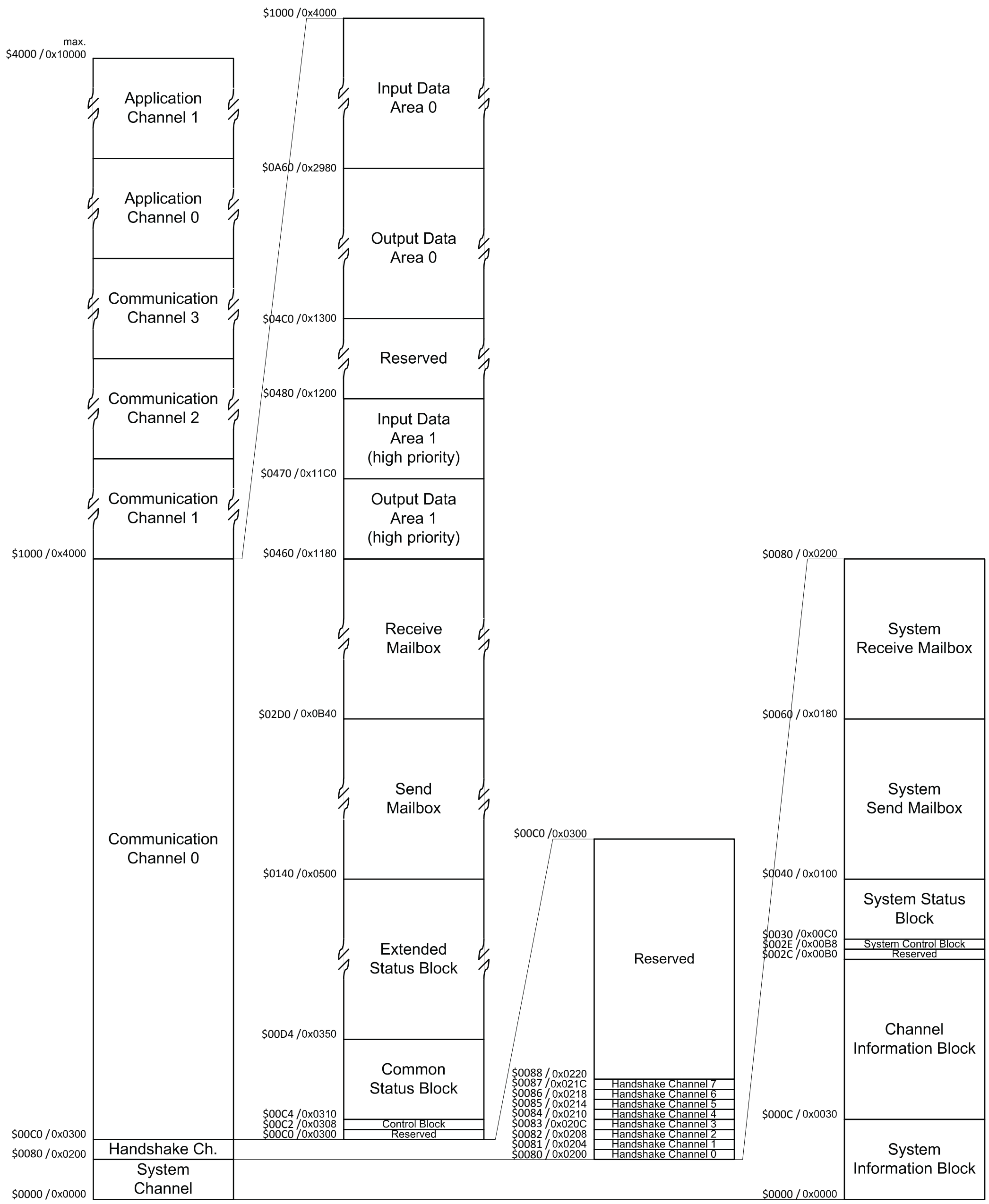
Note:

The USB connection is not a galvanically isolated connection. The ground of the PC will be connected to the ground of the UMAC system through the USB connection, which can damage components on the PC and/or ACC-72EX. Make sure that there is no potential difference between the grounds on both ends.

DPRAM MEMORY MAP

Below is the standard memory map of address offsets found in the DPRAM of the ACC-72EX module. Start and end addresses for each of the memory blocks are specified both in Hilscher offset (0x notation for hexadecimal) and Turbo PMAC offset ($ notation for hexadecimal) notation. The memory map shown here is the standard memory map. Different COMX modules may have different memory maps. Please refer to the Hilscher manual for each COMX module for detailed information.

These registers should be read from and written to using M-Variables which point to the lower 16 bits of the X/Y-memory with an offset from the base address that is configured with E3. Handshake and System registers are common between all protocols, and others can be auto-generated using the “ACC-72EX Setup Assistant” software.



DPRAM Blocks

In the Hilscher COMX module DPRAM, the system channel and the handshake channel are always present. These channels are used for communicating with the firmware on the COMX module, from this point on referred to as “netX.”

The system channel provides information about the state of the COMX module operating system, netX, and the structure of the dual-port memory. It allows basic communication via system mailboxes.

The handshake channel provides a bit toggle mechanism that insures synchronizing data transfer between the UMAC and COMX module. All handshake cells from system, communication, and application channels are brought together in this one location.

Next are the communication and application channels. A communication channel provides network access and occupies an area of the COMX dual-port memory with process, non-cyclic, and diagnostic data. An application channel can be used for any functionality that may be executed in the context of the netX operating system. The application channels are not supported by COMX modules at the time of writing this manual.

DPRAM Suggested Macro Names

ACC-72EX Setup Assistant software, available through the Tools menu in PEWIN32PRO2 software, provides a complete overview of blocks and sub-blocks available to each ACC-72EX. Under each section is a list of macro names provided for M-Variable definition.

For more information on structures and data registers in COMX modules, please refer to references introduced in appendix A of this manual.

System Channel

The System Channel is the first of the channels in the dual-port memory and starts at address offset $0000. It holds information about the system itself (netX, netX operating system) and provides a mailbox transfer mechanism for system-related messages or packets.

ACC-72EX Setup Assistant software uses the data available in this channel to generate the information in the memory map output file.

System Information Block

The first block of information allows identification of the netX dual memory; it is used for testing proper communication. The first 4 registers hold character values for “netX” (110, 101, 116, 88). If these values are reading properly, the DPRAM communication is in working condition.

	Hilscher Documentation	ACC-72EX Setup Assistant
System Information Block	abCookie[4]	SI_abCookie_0_ .. SI_abCookie_3_
	ulDpmTotalSize	SI_ulDpmTotalSize
	ulDeviceNumber	SI_ulDeviceNumber
	ulSerialNumber	SI_ulSerialNumber
	ausHwOptions[4]	SI_ausHwOptions_0_ .. SI_ausHwOptions_3_
	usManufacturer	SI_usManufacturer
	usProductionDate	SI_usProductionDate
	ulLicenseFlags1	SI_ulLicenseFlags1
	ulLicenseFlags2	SI_ulLicenseFlags2
	usNetxLicenseID	SI_usNetxLicenseID
	usNetxLicenseFlags	SI_usNetxLicenseFlags
	usDeviceClass	SI_usDeviceClass
	bHwRevision	SI_bHwRevision
	bHwCompatibility	SI_bHwCompatibility
	bDevIdNumber	SI_bDevIdNumber

Channel Information Block

The system block includes information about all the other channels and their availability on the COMX module. This information is used to locate and identify different channels in the system.

	Hilscher Documentation	ACC-72EX Setup Assistant
System Channel Information	bChannelType	SCI_bChannelType
	bSizePositionOfHandshake	SCI_bSizePositionOfHandshake
	bNumberOfBlocks	SCI_bNumberOfBlocks
	ulSizeOfChannel	SCI_ulSizeOfChannel
	usSizeOfMailbox	SCI_usSizeOfMailbox
	usMailboxStartOffset	SCI_usMailboxStartOffset

	Hilscher Documentation	ACC-72EX Setup Assistant
Handshake Channel Info.	bChannelType	HCI_bChannelType
	ulSizeOfChannel	HCI_ulSizeOfChannel

	Hilscher Documentation	ACC-72EX Setup Assistant
Communication Channel Information	bChannelType	CCxI_bChannelType
	bChannelId	CCxI_bChannelId
	bSizePositionOfHandshake	CCxI_bSizePositionOfHandshake
	bNumberOfBlocks	CCxI_bNumberOfBlocks
	ulSizeOfChannel	CCxI_ulSizeOfChannel
	usCommunicationClass	CCxI_usCommunicationClass
	usProtocolClass	CCxI_usProtocolClass
	usConformanceClass	CCxI_usConformanceClass
Note: x in MACRO name is replaced by Application Channel number 0 … 3		

	Hilscher Documentation	ACC-72EX Setup Assistant
Application Channel Info.	bChannelType	ACxI_bChannelType
	bChannelId	ACxI_bChannelId
	bSizePositionOfHandshake	ACxI_bSizePositionOfHandshake
	bNumberOfBlocks	ACxI_bNumberOfBlocks
	ulSizeOfChannel	ACxI_ulSizeOfChannel
Note: x in MACRO name is replaced by Application Channel number 0 … 1		

System Control Block

The system control block is used by UMAC to force netX to execute certain commands in the future. Currently, there are no such commands defined.

	Hilscher Documentation	ACC-72EX Setup Assistant
System Control Block	ulSystemCommandCOS	SCtrl_ulSystemCommandCOS

System Status Block

The system status block provides information about the staus of the netX firmware.

	Hilscher Documentation	ACC-72EX Setup Assistant
System Status Block	ulSystemCOS	SStat_ulSystemCOS
	ulSystemStatus	SStat_ulSystemStatus
	ulSystemError	SStat_ulSystemError
	ulBootError	SStat_ulBootError
	ulTimeSinceStart	SStat_ulTimeSinceStart
	usCpuLoad	SStat_usCpuLoad
	ulHWFeatures	SStat_ulHWFeatures

System Mailbox

The system mailbox is the "window" to the operating system. It is always present even if no firmware is loaded. For more information about using system send/receive mailboxes, please see the examples shown in the following chapters. A complete list of functions, which can be accessed using the mailboxes, can be found in the netX Dual-Ported Memory Interface document available from Hilscher.

	Hilscher Documentation	ACC-72EX Setup Assistant
System Block Send Mailbox	usPackagesAccepted	SSMB_usPackagesAccepted
	ulDest	SSMB_ulDest
	ulSrc	SSMB_ulSrc
	ulDestId	SSMB_ulDestId
	ulSrcId	SSMB_ulSrcId
	ulLen	SSMB_ulLen
	ulId	SSMB_ulId
	ulState	SSMB_ulState
	ulCmd	SSMB_ulCmd
	ulExt	SSMB_ulExt
	ulRout	SSMB_ulRout
	…	SSMB_ultData0 .. SSMB_ultData20

	Hilscher Documentation	ACC-72EX Setup Assistant
System Block Receive Mailbox	usWaitingPackages	SRMB_usWaitingPackages
	ulDest	SRMB_ulDest
	ulSrc	SRMB_ulSrc
	ulDestId	SRMB_ulDestId
	ulSrcId	SRMB_ulSrcId
	ulLen	SRMB_ulLen
	ulId	SRMB_ulId
	ulState	SRMB_ulState
	ulCmd	SRMB_ulCmd
	ulExt	SRMB_ulExt
	ulRout	SRMB_ulRout
	…	SRMB_ultData0 .. SRMB_ultData20

Handshake Channel

The handshake channel provides a mechanism that allows the synchronizing of data transfer between the UMAC CPU and ACC-72EX dual-port memory. The handshake channel brings all handshake registers from other channel blocks together in one location. The handshake register could be moved from the handshake block to the beginning of each of the communication channels.

There are three types of handshake cells, described below.

System Handshake Cells

System handshake flags are used to synchronize data transfer between the ACC-72EX Hilscher Module and UMAC via the system mailbox and to handle certain changes of state function. They also hold information about the status of the ACC-72EX Hilscher module and can be used to execute certain commands in the module (for a module-wide reset, for example).

There are two sets of system flags. One set is dedicated for netX writes and is read by UMAC, and the other one is designated for UMAC writes. netX is continuously reading the second set.

netX System Flags

The ACC-72EX Hilscher module firmware writes to the netX system register; UMAC reads this register. The netX system register is located at address offset $80 in the dual-port memory.

bNetxFlags – netX writes, UMAC reads

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	Reserved										NSF_RECV_MBX_CMD	NSF_SEND_MBX_ACK	NSF_NETX_COS_CMD	NSF_HOST_COS_ACK	NSF_ERROR	NSF_READY

netX System Flags **bNetxFlags** (ACC-72EX ⇨ UMAC)

Bit No.	Definition / Description
0	Ready (NSF_READY) The Ready flag is set as soon as the COMX has initialized itself properly and passed its self-test. When the flag is set, the netX is ready to accept packets via the system mailbox. If cleared, the netX does not accept any packages.
1	Error (NSF_ERROR) The Error flag is set when the netX has detected an internal error condition. This is considered to be a fatal error. The Ready flag is cleared and the netX operating system is stopped. An error code helping to identify the issue is placed in the ulSystemError variable in the system status block.
2	Host Change Of State Acknowledge (NSF_HOST_COS_ACK) The Host Change of State Acknowledge flag is set when the netX acknowledges a command from the host system. This flag is used together with the Host Change of State Command flag in the host system flags.
3	netX Change Of State Command (NSF_NETX_COS_CMD) The netX Change of State Command flag is set if the netX signals a change of its state to the host system. Details of what has changed can be found in the ulSystemCOS variable in the system control block.
4	Send Mailbox Acknowledge (NSF_SEND_MBX_ACK) Both the Send Mailbox Acknowledge flag and the Send Mailbox Command flag are used together to transfer non-cyclic packages between the UMAC and the netX.
5	Receive Mailbox Command (NSF_RECV_MBX_CMD) Both the Receive Mailbox Command flag and the Receive Mailbox Acknowledge flag are used together to transfer non-cyclic packages between the netX and UMAC.
6, 7 … 15	6, 7 … 15 Reserved, set to zero

	Hilscher Documentation	ACC-72EX Setup Assistant
System Handshake Register netX System Flags	bNetxFlags	HCSC_bNetxFlags
	NSF_READY	HCSC_NSF_READY
	NSF_ERROR	HCSC_NSF_ERROR
	NSF_HOST_COS_ACK	HCSC_NSF_HOST_COS_ACK
	NSF_NETX_COS_CMD	HCSC_NSF_NETX_COS_CMD
	NSF_SEND_MBX_ACK	HCSC_NSF_SEND_MBX_ACK
	NSF_RECV_MBX_CMD	HCSC_NSF_RECV_MBX_CMD

Host System Flags

The host system flags are written by UMAC; the netX reads these flags. The host system register is located at address offset $81 in the dual-port memory.

bHostFlags – UMAC writes, netX reads

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	Reserved										HSF_RECV_MBX_ACK	HSF_SEND_MBX_CMD	HSF_NETX_COS_ACK	HSF_HOST_COS_CMD	HSF_BOOTSTART	HSF_RESET

Host System Flags **bHostFlags** (UMAC ⇨ ACC-72EX)

Bit No.	Definition / Description
0	Reset (HSF_RESET) The Reset flag is set by the UMAC to execute a system wide reset. This forces the system to restart. All network connections are interrupted immediately regardless of their current state.
1	Bootstart (HSF_BOOTSTART) If set during reset, the Boot-Start flag forces the netX to stay in boot loader mode; a firmware that may reside in the context of the operating system rcX is not started. If cleared during reset, the operating system will start the firmware if available.
2	Host Change Of State Command (HSF_HOST_COS_CMD) The Host Change of State Command flag is set by the UMAC to signal a change of its state to the netX. Details of what has changed can be found in the ulSystemCommandCOS variable in the system control block.
3	netX Change Of State Acknowledge (HSF_NETX_COS_ACK) The netX Change of State Acknowledge flag is set by the UMAC to acknowledge the new state of the netX. This flag is used together with the netX Change of State Command flag in the netX system flags.
4	Send Mailbox Command (HSF_SEND_MBX_CMD) Both the Send Mailbox Command flag and the Send Mailbox Acknowledge flag are used together to transfer non-cyclic packages between the UMAC and the netX.
5	Receive Mailbox Acknowledge (HSF_RECV_MBX_ACK) Both the Receive Mailbox Acknowledge flag and the Receive Mailbox Command flag are used together to transfer non-cyclic packages between the netX and the UMAC.
6, 7 … 15	6, 7 … 15 Reserved; set to zero

	Hilscher Documentation	ACC-72EX Setup Assistant
System Handshake Register Host System Flags	bHostFlags	HCSC_bHostFlags
	HSF_RESET	HCSC_HSF_RESET
	HSF_BOOTSTART	HCSC_HSF_BOOTSTART
	HSF_HOST_COS_CMD	HCSC_HSF_HOST_COS_CMD
	HSF_NETX_COS_ACK	HCSC_HSF_NETX_COS_ACK
	HSF_SEND_MBX_CMD	HCSC_HSF_SEND_MBX_CMD
	HSF_RECV_MBX_ACK	HCSC_HSF_RECV_MBX_ACK

Communication Channel Handshake Cells

The channel handshake register is used to indicate the status of the protocol stack as well as execute certain commands in the protocol stack (e.g. reset a channel or synchronization of process data). The mailbox flags are used to send and receive non-cyclic messages via the channel mailboxes.

There are two sets of Communication Channel flags. One set is dedicated for netX writes; UMAC continually reads this. The other set is designated for UMAC writes; netX continuously reads this.

netX Communication Flags

This flag register is organized as a bit field. The netX protocol stack writes to the register to control data synchronization via the mailbox system and the process data image. It also informs the UMAC about its current network state. The UMAC reads this register.

usNetxFlags – netX writes, UMAC reads

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	Reserved						NCF_PD1_IN_CMD (not supported yet)	NCF_PD1_OUT_ACK (not supported yet)	NCF_PD0_IN_CMD	NCF_PD0_OUT_ACK	NCF_RECV_MBX_CMD	NCF_SEND_MBX_ACK	NCF_NETX_COS_CMD	NCF_HOST_COS_ACK	NCF_ERROR	NCF_COMMUNICATING

Communication Channel Flags **usNetXFlags** (ACC-72EX ⇨ UMAC)

Bit No.	Definition / Description
0	Communicating (NCF_COMMUNICATING) The NCF_COMMUNICATING flag is set if the protocol stack has successfully opened a connection to at least one of the configured network slaves (for master protocol stacks), respectively has an open connection to the network master (for slave protocol stacks). If cleared, the input data should not be evaluated, because it may be invalid, old or both. At initialization time, this flag is cleared.
1	Error (NCF_ERROR) The NCF_ERROR flag signals an error condition that is reported by the protocol stack. It could indicate a network communication issue or something to that effect. The corresponding error code is placed in the ulCommunicationError variable in the common status block. At initialization time, this flag is cleared.
2	Host Change Of State Acknowledge (NCF_HOST_COS_ACK) The NCF_HOST_COS_ACK flag is used by the protocol stack indicating that the new state of the UMAC has been read. At initialization time, this flag is cleared.
3	netX Change Of State Command (NCF_NETX_COS_CMD) The NCF_NETX_COS_CMD flag signals a change in the state of the protocol stack. The new state can be found in the ulCommunicationCOS register in the common status block. In return the UMAC program then toggles the HCF_NETX_COS_ACK flag in the host communication flags acknowledging that the new protocol state has been read. At initialization time, this flag is cleared.
4	Send Mailbox Acknowledge (NCF_SEND_MBX_ACK) Both the NCF_SEND_MBX_ACK flag and the HCF_SEND_MBX_CMD flag are used together to transfer non-cyclic packages between the protocol stack and the UMAC programs. At initialization time, this flag is cleared.
5	Receive Mailbox Command (NCF_RECV_MBX_CMD) Both the NCF_RECV_MBX_CMD flag and the HCF_RECV_MBX_ACK flag are used together to transfer non-cyclic packages between the UMAC programs and the protocol stack. At initialization time, this flag is cleared.
6	Process Data 0 Out Acknowledge (NCF_PD0_OUT_ACK) Both the NCF_PD0_OUT_ACK flag and the HCF_PD0_OUT_CMD flag are used together to transfer cyclic output data from the UMAC to the protocol stack. At initialization time, this flag may be set, depending on the data exchanged mode.
7	Process Data 0 In Command (NCF_PD0_IN_CMD) Both the NCF_PD0_IN_CMD flag and the HCF_PD0_IN_ACK flag are used together to transfer cyclic input data from the protocol stack to the UMAC. At initialization time, this flag may be set, depending on the data exchanged mode.
8	Process Data 1 Out Acknowledge (NCF_PD1_OUT_ACK, not supported yet) Both the NCF_PD1_OUT_ACK flag and the HCF_PD1_OUT_CMD flag are used together to transfer output cyclic data from the UMAC to the protocol stack. At initialization time, this flag may be set, depending on the data exchanged mode.
9	Process Data 1 In Command (NCF_PD1_IN_CMD, not supported yet) Both the NCF_PD1_IN_CMD flag and the HCF_PD1_IN_ACK flag are used together to transfer cyclic input data from the protocol stack to the UMAC. At initialization time, this flag may be set, depending on the data exchange mode.
10..15	Reserved, set to 0

	Hilscher Documentation	ACC-72EX Setup Assistant
netX Communication Flags Handshake Register	usNetxFlags	HCCCx_usNetxFlags
	NCF_COMMUNICATING	HCCCx_NCF_COMMUNICATING
	NCF_ERROR	HCCCx_NCF_ERROR
	NCF_HOST_COS_ACK	HCCCx_NCF_HOST_COS_ACK
	NCF_NETX_COS_CMD	HCCCx_NCF_NETX_COS_CMD
	NCF_SEND_MBX_ACK	HCCCx_NCF_SEND_MBX_ACK
	NCF_RECV_MBX_CMD	HCCCx_NCF_RECV_MBX_CMD
	NCF_PD0_OUT_ACK	HCCCx_NCF_PDx_OUT_ACK
	NCF_PD0_IN_CMD	HCCCx_NCF_PDx_IN_CMD
	NCF_PD1_OUT_ACK	HCCCx_NCF_PD1_OUT_ACK
	NCF_PD1_IN_CMD	HCCCx_NCF_PD1_IN_CMD
Note: x in MACRO name is replaced by Communication Channel number 0 .. 3		

Host Communication Flags

This flag register is organized as a bit field. UMAC writes to this register to control data synchronization via the mailbox system and the process data image. The netX protocol stack reads this register.

usHostFlags – UMAC writes, netX reads

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	Reserved						HCF_PD1_IN_ACK (not supported yet)	HCF_PD1_OUT_CMD (not supported yet)	HCF_PD0_IN_ACK	HCF_PD0_OUT_CMD	HCF_RECV_MBX_ACK	HCF_SEND_MBX_CMD	HCF_NETX_COS_ACK	HCF_HOST_COS_CMD	unused	

Communication Channel Flags **usHostFlags** (UMAC ⇨ ACC-72EX)

Bit No.	Definition / Description
0, 1	Reserved, set to 0
2	Host Change Of State Command (HCF_HOST_COS_CMD) The HCF_HOST_COS_CMD flag signals a change in the state of the UMAC. A new state is set in the ulApplicationCOS variable in the communication control block. The protocol stack on the netX then toggles the NCF_HOST_COS_ACK flag in the netX communication flags back acknowledging that the new state has been read. At initialization time, this flag is cleared.
3	Host Change Of State Acknowledge (HCF_NETX_COS_ACK) The HCF_NETX_COS_ACK flag is used by UMAC to indicate that the new state of the protocol stack has been read. At initialization time, this flag is cleared.
4	Send Mailbox Command (HCF_SEND_MBX_CMD) Both the HCF_SEND_MBX_CMD flag and the NCF_SEND_MBX_ACK flag are used together to transfer non-cyclic packages between the UMAC and the protocol stack. At initialization time, this flag is cleared.
5	Receive Mailbox Acknowledge (HCF_RECV_MBX_ACK) Both the HCF_RECV_MBX_ACK flag and the NCF_RECV_MBX_CMD flag are used together to transfer non-cyclic packages between the protocol stack and the UMAC. At initialization time, this flag is cleared.
6	Process Data 0 Out Command (HCF_PD0_OUT_CMD) Both the HCF_PD0_OUT_CMD flag and the NCF_PD0_OUT_ACK flag are used together to transfer cyclic output data from the UMAC to the protocol stack. At initialization time, this flag may be set, depending on the data exchanged mode.
7	Process Data 0 In Acknowledge (HCF_PD0_IN_ACK) Both the HCF_PD0_IN_ACK flag and the NCF_PD0_IN_CMD flag are used together to transfer cyclic input data from the protocol stack to the UMAC. At initialization time, this flag may be set, depending on the data exchanged mode.
8	Process Data 1 Out Command (HCF_PD1_OUT_CMD, not supported yet) Both the HCF_PD1_OUT_CMD flag and the NCF_PD1_OUT_ACK flag are used together to transfer cyclic output data from the UMAC to the protocol stack. At initialization time, this flag may be set, depending on the data exchanged mode.
9	Process Data 1 In Acknowledge (HCF_PD1_IN_ACK, not supported yet) Both the HCF_PD1_IN_ACK flag and the NCF_PD1_IN_CMD flag are used together to transfer cyclic input data from the protocol stack to the UMAC. At initialization time, this flag may be set, depending on the data exchanged mode.
10 ... 15	Reserved, set to 0

	Hilscher Documentation	ACC-72EX Setup Assistant
Host Communication Flags Handshake Register	usHostFlags	HCCCx_usHostFlags
	HCF_HOST_COS_CMD	HCCCx_HCF_HOST_COS_CMD
	HCF_NETX_COS_ACK	HCCCx_HCF_NETX_COS_ACK
	HCF_SEND_MBX_CMD	HCCCx_HCF_SEND_MBX_CMD
	HCF_RECV_MBX_ACK	HCCCx_HCF_RECV_MBX_ACK
	HCF_PD0_OUT_CMD	HCCCx_HCF_PDx_OUT_CMD
	HCF_PD0_IN_ACK	HCCCx_HCF_PDx_IN_ACK
	HCF_PD1_OUT_CMD	HCCCx_HCF_PD1_OUT_CMD
	HCF_PD1_IN_ACK	HCCCx_HCF_PD1_IN_ACK
Note: x in MACRO name is replaced by Communication Channel number 0 .. 3		

Application Handshake Cells

Although these cells are not supported yet, the following structure groups have been defined for backward compatibility as a placeholder:

netX Communication Flags

Host Communication Flags

Communication Channel

The communication channel structure is mainly dependent on the protocol firmware and COMX module. However, there are common sub-block structures which are common to all protocols.

Control Block

The control block of a dual-port memory features a Watchdog function to allow the operating system running on the netX to supervise the host application and vice versa. The control area is always present in dual-port memory. This block can also be read using the mailbox interface.

Application Change of State Register

The Application Change of State Register is a bit field. The UMAC uses this field to send commands to the communication channel. Changing flags in this register requires the UMAC to toggle the Host Change of State Command flag in the Host Communication Flags register, and then the netX protocol stack will recognize the change.

ulApplicationCOS – UMAC writes, netX reads

Bit	31	30	29	...	11	10	9	8	7	6	5	4	3	2	1	0
Function	Reserved							RCX_APP_COS_DMA_ENABLE	RCX_APP_COS_DMA	RCX_APP_COS_LOCK_CONFIG_ENABLE	RCX_APP_COS_LOCK_CONFIG	RCX_APP_COS_INIT_ENABLE	RCX_APP_COS_INIT	RCX_APP_COS_BUS_ON_ENABLE	RCX_APP_COS_BUS_ON	RCX_APP_COS_APP_READY

Bit No.	Definition / Description
0	Application Ready (RCX_APP_COS_APP_READY, not supported yet) If set, the UMAC indicates to the protocol stack that its state is Ready.
1	Bus On (RCX_APP_COS_BUS_ON) Using the Bus On flag, the UMAC allows or disallows the firmware to open network connections. This flag is used with Bus On Enable flag below. If set, the netX firmware tries to open network connections; if cleared, no connections are allowed, and open connections are closed.
2	Bus On Enable (RCX_APP_COS_BUS_ON_ENABLE) The Bus On Enable flag is used together with the Bus On flag above. If set, this flag enables the execution of the Bus On command in the netX firmware.
3	Initialization (RCX_APP_COS_INIT) Setting the Initialization flag the UMAC forces the protocol stack to restart and evaluate the configuration parameter again. All network connections are interrupted immediately regardless of their current state. If the database is locked, re-initializing the channel is not allowed.
4	Initialization Enable (RCX_APP_COS_INIT_ENABLE) The Initialization Enable flag is used together with the Initialization flag above. If set, this flag enables the execution of the Initialization command in the netX firmware.
5	Lock Configuration (RCX_APP_COS_LOCK_CONFIG) If set, UMAC does not allow the firmware to reconfigure the communication channel. The database will be locked. The Configuration Locked flag in the channel status block shows if the current database has been locked.

Bit No.	Definition / Description
6	Lock Configuration Enable (RCX_APP_COS_LOCK_CONFIG_ENABLE) The Lock Configuration Enable flag is used together with the Lock Configuration flag above. If set, this flag enables the execution of the Lock Configuration command in the netX firmware.
7	Turn on DMA Mode (RCX_APP_COS_DMA) The UMAC sets this flag in order to turn on the DMA mode for the cyclic process data input / output image 0 (abPd0Output and abPd0Input).
8	Turn on DMA Mode Enable (RCX_APP_COS_DMA_ENABLE) The DMA Enable flag is used together with the DMA flag above. If set, this flag enables the execution of the DMA command in the netX firmware.
9 ... 31	Reserved, set to 0

Device Watchdog Register

The protocol stack supervises the UMAC using a Watchdog function. If the UMAC fails to copy the value from the host Watchdog location to the device Watchdog location, the protocol stack assumes that the UMAC system has a problem and interrupts all network connections immediately, regardless of their current state.

	Hilscher Documentation	ACC-72EX Setup Assistant
Communication Control Block	RCX_APP_COS_APP_READY	CCx_RCX_APP_COS_APP_READY
	RCX_APP_COS_BUS_ON	CCx_RCX_APP_COS_BUS_ON
	RCX_APP_COS_BUS_ON_ENABLE	CCx_RCX_APP_COS_BUS_ON_ENABLE
	RCX_APP_COS_INIT	CCx_RCX_APP_COS_INIT
	RCX_APP_COS_INIT_ENABLE	CCx_RCX_APP_COS_INIT_ENABLE
	RCX_APP_COS_LOCK_CFG	CCx_RCX_APP_COS_LOCK_CFG
	RCX_APP_COS_LOCK_CFG_ENA	CCx_RCX_APP_COS_LOCK_CFG_ENA
	RCX_APP_COS_DMA	CCx_RCX_APP_COS_DMA
	RCX_APP_COS_DMA_ENABLE	CCx_RCX_APP_COS_DMA_ENABLE
	ulDeviceWatchdog	CCx_ulDeviceWatchdog
Note: x in MACRO name is replaced by Application Channel number 0 ... 3		

Common Status Block

The common status block contains information fields that are common to all protocol stacks. The status block is always present in dual-port memory. This block can also be read using the mailbox interface.

Communication Change of State Register

The Communication Change of State register is a bit field. It contains information about the current operating status of the communication channel and its firmware. Every time the status changes, the netX protocol stack toggles the netX Change of State Command flag in the netX communication flags register. The UMAC then has to toggle the netX Change of State Acknowledge flag back, acknowledging the new state.

ulCommunicationCOS - netX writes, UMAC reads

Bit	31	30	29	...	11	10	9	8	7	6	5	4	3	2	1	0
Function	Reserved								RCX_COMM_COS_DMA	RCX_COMM_COS_RESTART_REQUIRED_ENABLE	RCX_COMM_COS_RESTART_REQUIRED	RCX_COMM_COS_CONFIG_NEW	RCX_COMM_COS_CONFIG_LOCKED	RCX_COMM_COS_BUS_ON	RCX_COMM_COS_RUN	RCX_COMM_COS_READY

Bit No.	Definition / Description
0	Ready (RCX_COMM_COS_READY) The Ready flag is set as soon as the protocol stack is started properly. Then, the protocol stack awaits a configuration. As soon as the protocol stack is configured properly, the Running flag is set.
1	Running (RCX_COMM_COS_RUN) The Running flag is set when the protocol stack has been configured properly. Then the protocol stack awaits a network connection. Now, both the Ready flag and the Running flag are set.
2	Bus On (RCX_COMM_COS_BUS_ON) The Bus On flag is set to indicate to the UMAC whether or not the protocol stack has the permission to open network connections. If set, the protocol stack has the permission to communicate on the network; if cleared, the permission was denied and the protocol stack will not open network connections.
3	Configuration Locked (RCX_COMM_COS_CONFIG_LOCKED) The Configuration Locked flag is set if the communication channel firmware has locked the configuration database against being overwritten. Reinitializing the channel is not allowed in this state. To unlock the database, the application has to clear the Lock Configuration flag in the control block.
4	Configuration New (RCX_COMM_COS_CONFIG_NEW) The Configuration New flag is set by the protocol stack to indicate that a new configuration became available, but has not yet been activated. This flag may be set together with the Restart Required flag.
5	Restart Required (RCX_COMM_COS_RESTART_REQUIRED) The Restart Required flag is set when the channel firmware requests to be restarted. This flag is used together with the Restart Required Enable flag below. Restarting the channel firmware may become necessary if a new configuration was downloaded from the UMAC or if a configuration upload via the network took place.
6	Restart Required Enable (RCX_COMM_COS_RESTART_REQUIRED_ENABLE) The Restart Required Enable flag is used together with the Restart Required flag above. If set, this flag enables the execution of the Restart Required command in the netX firmware.
7	DMA Mode On (RCX_COMM_COS_DMA) The protocol stack sets this flag in order to signal to the UMAC that the DMA mode is turned on.
8 … 31	Reserved, set to 0

Communication State

The communication state field contains current device network communication status information. Depending on the implementation, all or a subset of the definitions below is supported:

Value	Definition / Description
$0	UNKNOWN
$1	OFFLINE
$2	STOP
$3	IDLE
$4	OPERATE

Communication Channel Error

This field holds the current error code of the communication channel. If the cause of error is resolved, the communication error field is set to zero (= RCX_S_OK) again. Not all of the error codes are supported in every implementation.

Watchdog Timeout

This field holds the configured Watchdog timeout value in milliseconds. The UMAC may set its Watchdog trigger interval accordingly. If the UMAC fails to copy the value from the host Watchdog location to the device Watchdog location, the protocol stack will interrupt all network connections immediately, regardless of their current state.

Handshake Mode

The protocol stack supports different handshake mechanisms to synchronize process data exchange with the UMAC. Depending on the configured mode, this mechanism insures data consistency over the entire data image and helps synchronize the UMAC with the network. This register holds the configured handshake mode.

Value	Definition / Description
$0	For compatibility reasons, this value is identical to 0x04 - Buffered Host Controlled IO Data Transfer
$2	Buffered Device-Controlled I/O Data Transfer
$3	Uncontrolled Mode
$4	Buffered Host-Controlled IO Data Transfer

Host Watchdog

The protocol stack supervises the UMAC via the Watchdog function. If the UMAC fails to copy the value from the device Watchdog location to the host Watchdog location, the protocol stack assumes that the UMAC has a problem and shuts down all network connections.

Error Count (All Implementations)

This field holds the total number of errors detected since power-up or after a reset. The protocol stack counts all sorts of errors in this field regardless if they were network-related or caused internally. The counter is cleared after a power cycle, reset, or channel initialization.

Error Log Indicator (All Implementations)

Not supported yet; the error log indicator field holds the number of entries in the internal error log. The field is set to zero if all entries are read from the log.

Number of Input Process Data Handshake Errors

TBD

Number of Output Process Data Handshake Errors

TBD

Number of Synchronization Handshake Errors

This counter will be incremented if the device detects a “not handled synchronization indication.” This field is not supported yet.

Synchronization Status

This field is reserved for future use.

Slave State

The Slave State field indicates whether or not the master is in cyclic data exchange to all configured slaves. If there is at least one slave missing or if the slave has a diagnostic request pending, the status

changes to FAILED. For protocols that support non-cyclic communication only, the slave state is set to OK as soon as a valid configuration is found.

Value	Definition / Description
$0	UNDEFINED
$1	OK. No Fault.
$2	FAILED. At least one slave failed
Other values are reserved	

Slave Error Log Indicator

Not supported yet: the error log indicator field holds the number of entries in the internal error log. The field is set to zero if all entries are read from the log.

Number of Configured Slaves

The firmware maintains a list of slaves with which the master has to open a connection. This list is derived from the configuration database created by SYCON.net. This field holds the number of configured slaves.

Number of Active Slaves

The firmware maintains a list of slaves to which the master exchanges process data. This field holds the number of active slaves. Ideally, the number of active slaves is equal to the number of configured slaves. For certain fieldbus systems, it could be possible that a slave is shown as activated, but still has a problem (i.e. a diagnostic issue).

Number of Faulted Slaves

The firmware maintains a list of slaves that are missing on the network, although they may be configured, or are reporting a diagnostic issue. As long as those indications are pending and not serviced, the field holds a nonzero value. If no more diagnostic information is pending, the field is set to zero again.

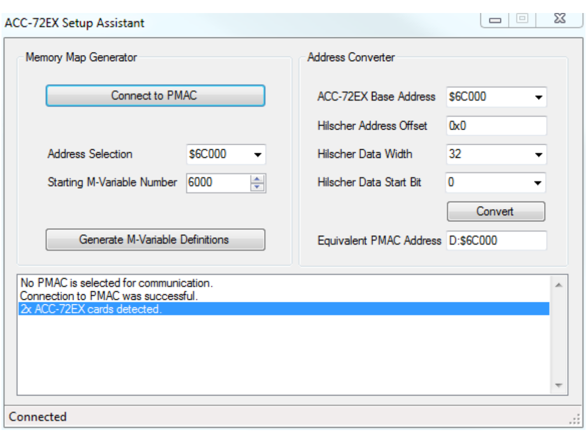
	Hilscher Documentation	ACC-72EX Setup Assistant
Common Status Block	RCX_COMM_COS_READY	CCx_RCX_COMM_COS_READY
	RCX_COMM_COS_RUN	CCx_RCX_COMM_COS_RUN
	RCX_COMM_COS_BUS_ON	CCx_RCX_COMM_COS_BUS_ON
	RCX_COMM_COS_CONFIG_LOCKED	CCx_RCX_COMM_COS_CONFIG_LOCKED
	RCX_COMM_COS_CONFIG_NEW	CCx_RCX_COMM_COS_CONFIG_NEW
	RCX_COMM_COS_RESTART_REQ	CCx_RCX_COMM_COS_RESTART_REQ
	RCX_COMM_COS_RESTART_REQ_ENA	CCx_RCX_COMM_COS_RESTART_REQ_ENA
	RCX_COMM_COS_DMA	CCx_RCX_COMM_COS_DMA
	ulCommunicationState	CCx_ulCommunicationState
	ulCommunicationError	CCx_ulCommunicationError
	usVersion	CCx_usVersion
	usWatchdogTime	CCx_usWatchdogTime
	bPDInHskMode	CCx_bPDInHskMode
	bPDInSource	CCx_bPDInSource
	bPDOutHskMode	CCx_bPDOutHskMode
	bPDOutSource	CCx_bPDOutSource
	ulHostWatchdog	CCx_ulHostWatchdog
	ulErrorCount	CCx_ulErrorCount
	bErrorLogInd	CCx_bErrorLogInd
	bErrorPDInCnt	CCx_bErrorPDInCnt
	bErrorPDOutCnt	CCx_bErrorPDOutCnt
	bErrorSyncCnt	CCx_bErrorSyncCnt
	bSyncHskMode	CCx_bSyncHskMode
	bSyncSource	CCx_bSyncSource
	ulSlaveState	CCx_ulSlaveState
	ulSlaveErrLogInd	CCx_ulSlaveErrLogInd
	ulNumOfConfigSlaves	CCx_ulNumOfConfigSlaves
	ulNumOfActiveSlaves	CCx_ulNumOfActiveSlaves
	ulNumOfDiagSlaves	CCx_ulNumOfDiagSlaves
Note: x in MACRO name is replaced by Application Channel number 0 .. 3		

Application Channel

The application channel is reserved for user specific implementations. An application channel is not yet supported.

Auto-Generated Dual-Ported Memory Map

ACC-72EX Setup Assistant Software, designed for use with Turbo PMAC, provides some level of automation in the identification of Hilscher COMX modules by generating a memory map file, suggested M-Variable definitions for important registers, and appropriate macro names.



Address Converter

The Address Converter section of the software allows conversion of offset, bit, and width parameters to PMAC memory addresses based on Hilscher documentation.

Memory Map Generator

The Memory Map Generator section of the software identifies the ACC-72EX cards in a UMAC system and generates both a memory map as a text file and M-Variable definition file with proper addressing, both of which indicate the ACC-72EX-based address selection.

Reading the Memory Map Text File

The output file from the software is a text file which can be read with any text editor software. This file includes generic information about the card.

Below is an example output file. Please see the notes in the right column for more information on specific items.

<table>
<tr><th>HilscherMemoryMap_$6C000.txt File Content</th><th>Notes</th></tr>
<tr><td>Delta Tau Data Systems, Inc.
ACC-72EX Setup Assistant Auto-generated Memory Map
ACC-72EX Address: $6C000</td><td>Base address of the ACC-72EX selected in the Memory Map Generator section</td></tr>
<tr><td>netX Identification: netX</td><td>The identification cookie provided by the netX firmware</td></tr>
<tr><td>Dual-Port Memory Size: 65536 bytes
Device Number: 1532100
Serial Number: 21456
Hardware Assembly Options:
                   Port 0:   ETHERNET (internal Phy)
                   Port 1:   ETHERNET (internal Phy)
                   Port 2:   NOT CONNECTED
                   Port 3:   NOT CONNECTED
Hilscher Module Production Date: Week 18 of 2012
Hilscher Module License Information: (PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) 1 Master License
Tool License Information: (SYCON.net)
Device Class: COMX 100</td><td></td></tr>
<tr><td> + Block 0:
 | Channel Type: System
 | Size of Channel: 512 bytes
 | Channel Start Address: $6C000
 | Position of Handshake Cells: IN HANDSHAKE CHANNEL</td><td>Block information
For all blocks</td></tr>
<tr><td> | netX System Flags Adress: X:$6C080,0,8
 | Host System Flags Adress: X:$6C080,8,8
 | Size of Handshake Cells: 8 BITS</td><td>Calculates where the handshake registers are Located</td></tr>
<tr><td> | Size of Mailbox: 256 bytes
 | Mailbox Start address: $6C040
 | Number of Subblocks: 5
 |</td><td></td></tr>
<tr><td> |--- Subblock 0: COMMON STATUS
 | Size: 176 bytes
 | Start Offset: $6C000
 | Transfer Direction: IN - OUT (Bi-Directional)
 | Transfer Type: DPM (Dual-Port Memory)
 | Handshake Mode: UNCONTROLLED
 | Handshake Bit: 0
 |
 |--- Subblock 1: CONTROL
 | Size: 8 bytes
 | Start Offset: $6C02E
 | Transfer Direction: OUT (Host System to netX)
 | Transfer Type: DPM (Dual-Port Memory)
 | Handshake Mode: UNCONTROLLED
 | Handshake Bit: 0
 |
 |--- Subblock 2: COMMON STATUS</td><td>Lists all channels’ Sub-Blocks</td></tr>
</table>

```
|      Size:              64 bytes
|      Start Offset:      $6C030
|      Transfer Direction: IN (netX to Host System)
|      Transfer Type:     DPM (Dual-Port Memory)
|      Handshake Mode:    UNCONTROLLED
|      Handshake Bit:     0
|
|--- Subblock 3: MAILBOX
|      Size:              128 bytes
|      Start Offset:      $6C040
|      Transfer Direction: OUT (Host System to netX)
|      Transfer Type:     DPM (Dual-Port Memory)
|      Handshake Mode:    BUFFERED, HOST CONTROLLED
|      Handshake Bit:     4
|
|--- Subblock 4: MAILBOX
       Size:              128 bytes
       Start Offset:      $6C060
       Transfer Direction: IN (netX to Host System)
       Transfer Type:     DPM (Dual-Port Memory)
       Handshake Mode:    UNKNOWN
       Handshake Bit:     5


+ Block 1:
|   Channel Type:                Handshake
|   Size of Channel:             256 bytes
|   Channel Start Address:       $6C080


+ Block 2:
|   Channel Type:                Communication
|   Size of Channel:             15616 bytes
|   Channel Start Address:       $6C0C0
|   Position of Handshake Cells: IN HANDSHAKE CHANNEL
|   Size of Handshake Cells:     16 BITS
|   NetX Handshake Register:     Y:$6C082,0,16
|   Host Handshake Register:     X:$6C082,0,16
|   Communication Class:         SCANNER
|   Protocol Class:              IO-DEVICE
|   Conformance Class:           0
|   Number of Subblocks:         9
|
|--- Subblock 0: CONTROL
|      Size:              8 bytes
|      Start Offset:      $6C0C2
|      Transfer Direction: OUT (Host System to netX)
|      Transfer Type:     DPM (Dual-Port Memory)
|      Handshake Mode:    UNCONTROLLED
|      Handshake Bit:     0
|
|--- Subblock 1: COMMON STATUS
|      Size:              64 bytes
|      Start Offset:      $6C0C4
|      Transfer Direction: IN (netX to Host System)
|      Transfer Type:     DPM (Dual-Port Memory)
|      Handshake Mode:    UNCONTROLLED
|      Handshake Bit:     0
|
|--- Subblock 2: EXTENDED STATUS
|      Size:              432 bytes
|      Start Offset:      $6C0D4
|      Transfer Direction: IN (netX to Host System)
|      Transfer Type:     DPM (Dual-Port Memory)
|      Handshake Mode:    UNCONTROLLED
|      Handshake Bit:     0
|
|--- Subblock 3: MAILBOX
|      Size:              1600 bytes
|      Start Offset:      $6C140
|      Transfer Direction: OUT (Host System to netX)
```

```
|      Transfer Type:      DPM (Dual-Port Memory)
|      Handshake Mode:     BUFFERED, HOST CONTROLLED
|      Handshake Bit:      4
|
|--- Subblock 4: MAILBOX
|      Size:               1600 bytes
|      Start Offset:       $6C2D0
|      Transfer Direction: IN (netX to Host System)
|      Transfer Type:      DPM (Dual-Port Memory)
|      Handshake Mode:     UNKNOWN
|      Handshake Bit:      5
|
|--- Subblock 5: PROCESS DATA IMAGE
|      Size:               5760 bytes
|      Start Offset:       $6C4C0
|      Transfer Direction: OUT (Host System to netX)
|      Transfer Type:      DPM (Dual-Port Memory)
|      Handshake Mode:     BUFFERED, HOST CONTROLLED
|      Handshake Bit:      6
|
|--- Subblock 6: PROCESS DATA IMAGE
|      Size:               5760 bytes
|      Start Offset:       $6CA60
|      Transfer Direction: IN (netX to Host System)
|      Transfer Type:      DPM (Dual-Port Memory)
|      Handshake Mode:     BUFFERED, HOST CONTROLLED
|      Handshake Bit:      7
|
|--- Subblock 7: HIGH PRIORITY DATA IMAGE
|      Size:               64 bytes
|      Start Offset:       $6C460
|      Transfer Direction: OUT (Host System to netX)
|      Transfer Type:      DPM (Dual-Port Memory)
|      Handshake Mode:     BUFFERED, HOST CONTROLLED
|      Handshake Bit:      8
|
|--- Subblock 8: HIGH PRIORITY DATA IMAGE
       Size:               64 bytes
       Start Offset:       $6C470
       Transfer Direction: IN (netX to Host System)
       Transfer Type:      DPM (Dual-Port Memory)
       Handshake Mode:     BUFFERED, HOST CONTROLLED
       Handshake Bit:      9


+ Block 3:
|   Channel Type:                 Communication
|   Size of Channel:              15616 bytes
|   Channel Start Address:        $6D000
|   Position of Handshake Cells:  IN HANDSHAKE CHANNEL
|   Size of Handshake Cells:      16 BITS
|   NetX Handshake Register:      Y:$6C083,0,16
|   Host Handshake Register:      X:$6C083,0,16
|   Communication Class:          MESSAGING
|   Protocol Class:               UNDEFINED
|   Conformance Class:            0
|   Number of Subblocks:             9
|
|--- Subblock 0: CONTROL
|      Size:               8 bytes
|      Start Offset:       $6D002
|      Transfer Direction: OUT (Host System to netX)
|      Transfer Type:      DPM (Dual-Port Memory)
|      Handshake Mode:     UNCONTROLLED
|      Handshake Bit:      0
|
|--- Subblock 1: COMMON STATUS
|      Size:               64 bytes
|      Start Offset:       $6D004
|      Transfer Direction: IN (netX to Host System)
|      Transfer Type:      DPM (Dual-Port Memory)
```

```
|      Handshake Mode:     UNCONTROLLED
|      Handshake Bit:      0
|
|--- Subblock 2: EXTENDED STATUS
|      Size:               432 bytes
|      Start Offset:       $6D014
|      Transfer Direction: IN (netX to Host System)
|      Transfer Type:      DPM (Dual-Port Memory)
|      Handshake Mode:     UNCONTROLLED
|      Handshake Bit:      0
|
|--- Subblock 3: MAILBOX
|      Size:               1600 bytes
|      Start Offset:       $6D080
|      Transfer Direction: OUT (Host System to netX)
|      Transfer Type:      DPM (Dual-Port Memory)
|      Handshake Mode:     BUFFERED, HOST CONTROLLED
|      Handshake Bit:      4
|
|--- Subblock 4: MAILBOX
|      Size:               1600 bytes
|      Start Offset:       $6D210
|      Transfer Direction: IN (netX to Host System)
|      Transfer Type:      DPM (Dual-Port Memory)
|      Handshake Mode:     UNKNOWN
|      Handshake Bit:      5
|
|--- Subblock 5: PROCESS DATA IMAGE
|      Size:               5760 bytes
|      Start Offset:       $6D400
|      Transfer Direction: OUT (Host System to netX)
|      Transfer Type:      DPM (Dual-Port Memory)
|      Handshake Mode:     BUFFERED, HOST CONTROLLED
|      Handshake Bit:      6
|
|--- Subblock 6: PROCESS DATA IMAGE
|      Size:               5760 bytes
|      Start Offset:       $6D9A0
|      Transfer Direction: IN (netX to Host System)
|      Transfer Type:      DPM (Dual-Port Memory)
|      Handshake Mode:     BUFFERED, HOST CONTROLLED
|      Handshake Bit:      7
|
|--- Subblock 7: HIGH PRIORITY DATA IMAGE
|      Size:               64 bytes
|      Start Offset:       $6D3A0
|      Transfer Direction: OUT (Host System to netX)
|      Transfer Type:      DPM (Dual-Port Memory)
|      Handshake Mode:     BUFFERED, HOST CONTROLLED
|      Handshake Bit:      8
|
|--- Subblock 8: HIGH PRIORITY DATA IMAGE
       Size:               64 bytes
       Start Offset:       $6D3B0
       Transfer Direction: IN (netX to Host System)
       Transfer Type:      DPM (Dual-Port Memory)
       Handshake Mode:     BUFFERED, HOST CONTROLLED
       Handshake Bit:      9


+ Block 4:
|   Channel Type:                 Undefined
|   Size of Channel:              0 bytes
|   Channel Start Address:        $6DF40
|   Position of Handshake Cells: BEGINNING OF CHANNEL
|   Size of Handshake Cells:      NOT AVAILABLE
|   NetX Handshake Register:      X:$6DF40
|   Host Handshake Register:      X:$6DF40,8,0
|   Communication Class:          UNDEFINED
|   Protocol Class:               UNDEFINED
|   Conformance Class:            0
```

```
|   Number of Subblocks:            0
|

+ Block 5:
|   Channel Type:                Undefined
|   Size of Channel:             0 bytes
|   Channel Start Address:       $6DF40
|   Position of Handshake Cells: BEGINNING OF CHANNEL
|   Size of Handshake Cells:     NOT AVAILABLE
|   NetX Handshake Register:     X:$6DF40
|   Host Handshake Register:     X:$6DF40,8,0
|   Communication Class:         UNDEFINED
|   Protocol Class:              UNDEFINED
|   Conformance Class:           0
|   Number of Subblocks:            0
```

Suggested M-Variable Definition File

Here is a sample macro name and suggested M-Variable definition file.

MacroNameDefinition_$6C000.h File Content

```
#define SI_abCookie_0_                       M6000
#define SI_abCookie_1_                       M6001
#define SI_abCookie_2_                       M6002
#define SI_abCookie_3_                       M6003
#define SI_ulDpmTotalSize                    M6004
#define SI_ulDeviceNumber                    M6005
#define SI_ulSerialNumber                    M6006
#define SI_ausHwOptions_0_                   M6007
#define SI_ausHwOptions_1_                   M6008
#define SI_ausHwOptions_2_                   M6009
#define SI_ausHwOptions_3_                   M6010
#define SI_usManufacturer                    M6011
#define SI_usProductionDate                  M6012
#define SI_ulLicenseFlags1                   M6013
#define SI_ulLicenseFlags2                   M6014
#define SI_usNetxLicenseID                   M6015
#define SI_usNetxLicenseFlags                M6016
#define SI_usDeviceClass                     M6017
#define SI_bHwRevision                       M6018
#define SI_bHwCompatibility                  M6019
#define SI_bDevIdNumber                      M6020
#define SCI_bChannelType                     M6021
#define SCI_bSizePositionOfHandshake         M6022
#define SCI_bNumberOfBlocks                  M6023
#define SCI_ulSizeOfChannel                  M6024
#define SCI_usSizeOfMailbox                  M6025
#define SCI_usMailboxStartOffset             M6026
#define HCI_bChannelType                     M6027
#define HCI_ulSizeOfChannel                  M6028
#define CC0I_bChannelType                    M6029
#define CC0I_bChannelId                      M6030
#define CC0I_bSizePositionOfHandshake        M6031
#define CC0I_bNumberOfBlocks                 M6032
#define CC0I_ulSizeOfChannel                 M6033
#define CC0I_usCommunicationClass            M6034
#define CC0I_usProtocolClass                 M6035
#define CC0I_usConformanceClass              M6036
#define CC1I_bChannelType                    M6037
#define CC1I_bChannelId                      M6038
#define CC1I_bSizePositionOfHandshake        M6039
#define CC1I_bNumberOfBlocks                 M6040
#define CC1I_ulSizeOfChannel                 M6041
#define CC1I_usCommunicationClass            M6042
#define CC1I_usProtocolClass                 M6043
#define CC1I_usConformanceClass              M6044
#define CC2I_bChannelType                    M6045
#define CC2I_bChannelId                      M6046
#define CC2I_bSizePositionOfHandshake        M6047
#define CC2I_bNumberOfBlocks                 M6048
```

```
#define CC2I_ulSizeOfChannel              M6049
#define CC2I_usCommunicationClass         M6050
#define CC2I_usProtocolClass              M6051
#define CC2I_usConformanceClass           M6052
#define CC3I_bChannelType                 M6053
#define CC3I_bChannelId                   M6054
#define CC3I_bSizePositionOfHandshake     M6055
#define CC3I_bNumberOfBlocks              M6056
#define CC3I_ulSizeOfChannel              M6057
#define CC3I_usCommunicationClass         M6058
#define CC3I_usProtocolClass              M6059
#define CC3I_usConformanceClass           M6060
#define AC0I_bChannelType                 M6061
#define AC0I_bChannelId                   M6062
#define AC0I_bSizePositionOfHandshake     M6063
#define AC0I_bNumberOfBlocks              M6064
#define AC0I_ulSizeOfChannel              M6065
#define AC1I_bChannelType                 M6066
#define AC1I_bChannelId                   M6067
#define AC1I_bSizePositionOfHandshake     M6068
#define AC1I_bNumberOfBlocks              M6069
#define AC1I_ulSizeOfChannel              M6070
#define SCtrl_ulSystemCommandCOS          M6071
#define SStat_ulSystemCOS                 M6072
#define SStat_ulSystemStatus              M6073
#define SStat_ulSystemError               M6074
#define SStat_ulBootError                 M6075
#define SStat_ulTimeSinceStart            M6076
#define SStat_usCpuLoad                   M6077
#define SStat_ulHWFeatures                M6078
#define SSMB_usPackagesAccepted           M6079
#define SSMB_ulDest                       M6080
#define SSMB_ulSrc                        M6081
#define SSMB_ulDestId                     M6082
#define SSMB_ulSrcId                      M6083
#define SSMB_ulLen                        M6084
#define SSMB_ulId                         M6085
#define SSMB_ulState                      M6086
#define SSMB_ulCmd                        M6087
#define SSMB_ulExt                        M6088
#define SSMB_ulRout                       M6089
#define SSMB_ultData0                     M6090
#define SSMB_ultData1                     M6091
#define SSMB_ultData2                     M6092
#define SSMB_ultData3                     M6093
#define SSMB_ultData4                     M6094
#define SSMB_ultData5                     M6095
#define SSMB_ultData6                     M6096
#define SSMB_ultData7                     M6097
#define SSMB_ultData8                     M6098
#define SSMB_ultData9                     M6099
#define SSMB_ultData10                    M6100
#define SSMB_ultData11                    M6101
#define SSMB_ultData12                    M6102
#define SSMB_ultData13                    M6103
#define SSMB_ultData14                    M6104
#define SSMB_ultData15                    M6105
#define SSMB_ultData16                    M6106
#define SSMB_ultData17                    M6107
#define SSMB_ultData18                    M6108
#define SSMB_ultData19                    M6109
#define SSMB_ultData20                    M6110
#define SRMB_usWaitingPackages            M6111
#define SRMB_ulDest                       M6112
#define SRMB_ulSrc                        M6113
#define SRMB_ulDestId                     M6114
#define SRMB_ulSrcId                      M6115
#define SRMB_ulLen                        M6116
#define SRMB_ulId                         M6117
#define SRMB_ulState                      M6118
#define SRMB_ulCmd                        M6119
```

```
#define SRMB_ulExt                            M6120
#define SRMB_ulRout                           M6121
#define SRMB_ultData0                         M6122
#define SRMB_ultData1                         M6123
#define SRMB_ultData2                         M6124
#define SRMB_ultData3                         M6125
#define SRMB_ultData4                         M6126
#define SRMB_ultData5                         M6127
#define SRMB_ultData6                         M6128
#define SRMB_ultData7                         M6129
#define SRMB_ultData8                         M6130
#define SRMB_ultData9                         M6131
#define SRMB_ultData10                        M6132
#define SRMB_ultData11                        M6133
#define SRMB_ultData12                        M6134
#define SRMB_ultData13                        M6135
#define SRMB_ultData14                        M6136
#define SRMB_ultData15                        M6137
#define SRMB_ultData16                        M6138
#define SRMB_ultData17                        M6139
#define SRMB_ultData18                        M6140
#define SRMB_ultData19                        M6141
#define SRMB_ultData20                        M6142
#define HCSC_bNetxFlags                       M6143
#define HCSC_NSF_READY                        M6144
#define HCSC_NSF_ERROR                        M6145
#define HCSC_NSF_HOST_COS_ACK                 M6146
#define HCSC_NSF_NETX_COS_CMD                 M6147
#define HCSC_NSF_SEND_MBX_ACK                 M6148
#define HCSC_NSF_RECV_MBX_CMD                 M6149
#define HCSC_bHostFlags                       M6150
#define HCSC_HSF_RESET                        M6151
#define HCSC_HSF_BOOTSTART                    M6152
#define HCSC_HSF_HOST_COS_CMD                 M6153
#define HCSC_HSF_NETX_COS_ACK                 M6154
#define HCSC_HSF_SEND_MBX_CMD                 M6155
#define HCSC_HSF_RECV_MBX_ACK                 M6156
#define HCCC0_usNetxFlags                     M6157
#define HCCC0_NCF_COMMUNICATING               M6158
#define HCCC0_NCF_ERROR                       M6159
#define HCCC0_NCF_HOST_COS_ACK                M6160
#define HCCC0_NCF_NETX_COS_CMD                M6161
#define HCCC0_NCF_SEND_MBX_ACK                M6162
#define HCCC0_NCF_RECV_MBX_CMD                M6163
#define HCCC0_NCF_PD0_OUT_ACK                 M6164
#define HCCC0_NCF_PD0_IN_CMD                  M6165
#define HCCC0_NCF_PD1_OUT_ACK                 M6166
#define HCCC0_NCF_PD1_IN_CMD                  M6167
#define HCCC0_usHostFlags                     M6168
#define HCCC0_HCF_HOST_COS_CMD                M6169
#define HCCC0_HCF_NETX_COS_ACK                M6170
#define HCCC0_HCF_SEND_MBX_CMD                M6171
#define HCCC0_HCF_RECV_MBX_ACK                M6172
#define HCCC0_HCF_PD0_OUT_CMD                 M6173
#define HCCC0_HCF_PD0_IN_ACK                  M6174
#define HCCC0_HCF_PD1_OUT_CMD                 M6175
#define HCCC0_HCF_PD1_IN_ACK                  M6176
#define HCCC1_usNetxFlags                     M6177
#define HCCC1_NCF_COMMUNICATING               M6178
#define HCCC1_NCF_ERROR                       M6179
#define HCCC1_NCF_HOST_COS_ACK                M6180
#define HCCC1_NCF_NETX_COS_CMD                M6181
#define HCCC1_NCF_SEND_MBX_ACK                M6182
#define HCCC1_NCF_RECV_MBX_CMD                M6183
#define HCCC1_NCF_PD0_OUT_ACK                 M6184
#define HCCC1_NCF_PD0_IN_CMD                  M6185
#define HCCC1_NCF_PD1_OUT_ACK                 M6186
#define HCCC1_NCF_PD1_IN_CMD                  M6187
#define HCCC1_usHostFlags                     M6188
#define HCCC1_HCF_HOST_COS_CMD                M6189
#define HCCC1_HCF_NETX_COS_ACK                M6190
```

```
#define HCCC1_HCF_SEND_MBX_CMD              M6191
#define HCCC1_HCF_RECV_MBX_ACK              M6192
#define HCCC1_HCF_PD0_OUT_CMD               M6193
#define HCCC1_HCF_PD0_IN_ACK                M6194
#define HCCC1_HCF_PD1_OUT_CMD               M6195
#define HCCC1_HCF_PD1_IN_ACK                M6196
#define HCCC2_usNetxFlags                   M6197
#define HCCC2_NCF_COMMUNICATING             M6198
#define HCCC2_NCF_ERROR                     M6199
#define HCCC2_NCF_HOST_COS_ACK              M6200
#define HCCC2_NCF_NETX_COS_CMD              M6201
#define HCCC2_NCF_SEND_MBX_ACK              M6202
#define HCCC2_NCF_RECV_MBX_CMD              M6203
#define HCCC2_NCF_PD0_OUT_ACK               M6204
#define HCCC2_NCF_PD0_IN_CMD                M6205
#define HCCC2_NCF_PD1_OUT_ACK               M6206
#define HCCC2_NCF_PD1_IN_CMD                M6207
#define HCCC2_usHostFlags                   M6208
#define HCCC2_HCF_HOST_COS_CMD              M6209
#define HCCC2_HCF_NETX_COS_ACK              M6210
#define HCCC2_HCF_SEND_MBX_CMD              M6211
#define HCCC2_HCF_RECV_MBX_ACK              M6212
#define HCCC2_HCF_PD0_OUT_CMD               M6213
#define HCCC2_HCF_PD0_IN_ACK                M6214
#define HCCC2_HCF_PD1_OUT_CMD               M6215
#define HCCC2_HCF_PD1_IN_ACK                M6216
#define HCCC3_usNetxFlags                   M6217
#define HCCC3_NCF_COMMUNICATING             M6218
#define HCCC3_NCF_ERROR                     M6219
#define HCCC3_NCF_HOST_COS_ACK              M6220
#define HCCC3_NCF_NETX_COS_CMD              M6221
#define HCCC3_NCF_SEND_MBX_ACK              M6222
#define HCCC3_NCF_RECV_MBX_CMD              M6223
#define HCCC3_NCF_PD0_OUT_ACK               M6224
#define HCCC3_NCF_PD0_IN_CMD                M6225
#define HCCC3_NCF_PD1_OUT_ACK               M6226
#define HCCC3_NCF_PD1_IN_CMD                M6227
#define HCCC3_usHostFlags                   M6228
#define HCCC3_HCF_HOST_COS_CMD              M6229
#define HCCC3_HCF_NETX_COS_ACK              M6230
#define HCCC3_HCF_SEND_MBX_CMD              M6231
#define HCCC3_HCF_RECV_MBX_ACK              M6232
#define HCCC3_HCF_PD0_OUT_CMD               M6233
#define HCCC3_HCF_PD0_IN_ACK                M6234
#define HCCC3_HCF_PD1_OUT_CMD               M6235
#define HCCC3_HCF_PD1_IN_ACK                M6236
#define HCAC0_usNetxFlags                   M6237
#define HCAC0_NCF_COMMUNICATING             M6238
#define HCAC0_NCF_ERROR                     M6239
#define HCAC0_NCF_HOST_COS_ACK              M6240
#define HCAC0_NCF_NETX_COS_CMD              M6241
#define HCAC0_NCF_SEND_MBX_ACK              M6242
#define HCAC0_NCF_RECV_MBX_CMD              M6243
#define HCAC0_NCF_PD0_OUT_ACK               M6244
#define HCAC0_NCF_PD0_IN_CMD                M6245
#define HCAC0_NCF_PD1_OUT_ACK               M6246
#define HCAC0_NCF_PD1_IN_CMD                M6247
#define HCAC0_usHostFlags                   M6248
#define HCAC0_HCF_HOST_COS_CMD              M6249
#define HCAC0_HCF_NETX_COS_ACK              M6250
#define HCAC0_HCF_SEND_MBX_CMD              M6251
#define HCAC0_HCF_RECV_MBX_ACK              M6252
#define HCAC0_HCF_PD0_OUT_CMD               M6253
#define HCAC0_HCF_PD0_IN_ACK                M6254
#define HCAC0_HCF_PD1_OUT_CMD               M6255
#define HCAC0_HCF_PD1_IN_ACK                M6256
#define HCAC1_usNetxFlags                   M6257
#define HCAC1_NCF_COMMUNICATING             M6258
#define HCAC1_NCF_ERROR                     M6259
#define HCAC1_NCF_HOST_COS_ACK              M6260
#define HCAC1_NCF_NETX_COS_CMD              M6261
```

```
#define HCAC1_NCF_SEND_MBX_ACK              M6262
#define HCAC1_NCF_RECV_MBX_CMD              M6263
#define HCAC1_NCF_PD0_OUT_ACK               M6264
#define HCAC1_NCF_PD0_IN_CMD                M6265
#define HCAC1_NCF_PD1_OUT_ACK               M6266
#define HCAC1_NCF_PD1_IN_CMD                M6267
#define HCAC1_usHostFlags                   M6268
#define HCAC1_HCF_HOST_COS_CMD              M6269
#define HCAC1_HCF_NETX_COS_ACK              M6270
#define HCAC1_HCF_SEND_MBX_CMD              M6271
#define HCAC1_HCF_RECV_MBX_ACK              M6272
#define HCAC1_HCF_PD0_OUT_CMD               M6273
#define HCAC1_HCF_PD0_IN_ACK                M6274
#define HCAC1_HCF_PD1_OUT_CMD               M6275
#define HCAC1_HCF_PD1_IN_ACK                M6276
#define CC0_RCX_APP_COS_APP_READY           M6277
#define CC0_RCX_APP_COS_BUS_ON              M6278
#define CC0_RCX_APP_COS_BUS_ON_ENABLE       M6279
#define CC0_RCX_APP_COS_INIT                M6280
#define CC0_RCX_APP_COS_INIT_ENABLE         M6281
#define CC0_RCX_APP_COS_LOCK_CFG            M6282
#define CC0_RCX_APP_COS_LOCK_CFG_ENA        M6283
#define CC0_RCX_APP_COS_DMA                 M6284
#define CC0_RCX_APP_COS_DMA_ENABLE          M6285
#define CC0_ulDeviceWatchdog                M6286
#define CC0_RCX_COMM_COS_READY              M6287
#define CC0_RCX_COMM_COS_RUN                M6288
#define CC0_RCX_COMM_COS_BUS_ON             M6289
#define CC0_RCX_COMM_COS_CONFIG_LOCKED      M6290
#define CC0_RCX_COMM_COS_CONFIG_NEW         M6291
#define CC0_RCX_COMM_COS_RESTART_REQ        M6292
#define CC0_RCX_COMM_COS_RESTART_REQ_ENA    M6293
#define CC0_RCX_COMM_COS_DMA                M6294
#define CC0_ulCommunicationState            M6295
#define CC0_ulCommunicationError            M6296
#define CC0_usVersion                       M6297
#define CC0_usWatchdogTime                  M6298
#define CC0_bPDInHskMode                    M6299
#define CC0_bPDInSource                     M6300
#define CC0_bPDOutHskMode                   M6301
#define CC0_bPDOutSource                    M6302
#define CC0_ulHostWatchdog                  M6303
#define CC0_ulErrorCount                    M6304
#define CC0_bErrorLogInd                    M6305
#define CC0_bErrorPDInCnt                   M6306
#define CC0_bErrorPDOutCnt                  M6307
#define CC0_bErrorSyncCnt                   M6308
#define CC0_bSyncHskMode                    M6309
#define CC0_bSyncSource                     M6310
#define CC0_ulSlaveState                    M6311
#define CC0_ulSlaveErrLogInd                M6312
#define CC0_ulNumOfConfigSlaves             M6313
#define CC0_ulNumOfActiveSlaves             M6314
#define CC0_ulNumOfDiagSlaves               M6315
#define CC1_RCX_APP_COS_APP_READY           M6316
#define CC1_RCX_APP_COS_BUS_ON              M6317
#define CC1_RCX_APP_COS_BUS_ON_ENABLE       M6318
#define CC1_RCX_APP_COS_INIT                M6319
#define CC1_RCX_APP_COS_INIT_ENABLE         M6320
#define CC1_RCX_APP_COS_LOCK_CFG            M6321
#define CC1_RCX_APP_COS_LOCK_CFG_ENA        M6322
#define CC1_RCX_APP_COS_DMA                 M6323
#define CC1_RCX_APP_COS_DMA_ENABLE          M6324
#define CC1_ulDeviceWatchdog                M6325
#define CC1_RCX_COMM_COS_READY              M6326
#define CC1_RCX_COMM_COS_RUN                M6327
#define CC1_RCX_COMM_COS_BUS_ON             M6328
#define CC1_RCX_COMM_COS_CONFIG_LOCKED      M6329
#define CC1_RCX_COMM_COS_CONFIG_NEW         M6330
#define CC1_RCX_COMM_COS_RESTART_REQ        M6331
#define CC1_RCX_COMM_COS_RESTART_REQ_ENA    M6332
```

```
#define CC1_RCX_COMM_COS_DMA              M6333
#define CC1_ulCommunicationState          M6334
#define CC1_ulCommunicationError          M6335
#define CC1_usVersion                     M6336
#define CC1_usWatchdogTime                M6337
#define CC1_bPDInHskMode                  M6338
#define CC1_bPDInSource                   M6339
#define CC1_bPDOutHskMode                 M6340
#define CC1_bPDOutSource                  M6341
#define CC1_ulHostWatchdog                M6342
#define CC1_ulErrorCount                  M6343
#define CC1_bErrorLogInd                  M6344
#define CC1_bErrorPDInCnt                 M6345
#define CC1_bErrorPDOutCnt                M6346
#define CC1_bErrorSyncCnt                 M6347
#define CC1_bSyncHskMode                  M6348
#define CC1_bSyncSource                   M6349
#define CC1_ulSlaveState                  M6350
#define CC1_ulSlaveErrLogInd              M6351
#define CC1_ulNumOfConfigSlaves           M6352
#define CC1_ulNumOfActiveSlaves           M6353
#define CC1_ulNumOfDiagSlaves             M6354
```

MacroNameDefinition_$6C000.pmc File Content

```
CLOSE
END GAT
DEL GAT
#Include "MacroNameDefinition_$6C000.h"

SI_abCookie_0_->Y:$6C000,0,8
SI_abCookie_1_->Y:$6C000,8,8
SI_abCookie_2_->X:$6C000,0,8
SI_abCookie_3_->X:$6C000,8,8
SI_ulDpmTotalSize->DP:$6C001
SI_ulDeviceNumber->DP:$6C002
SI_ulSerialNumber->DP:$6C003
SI_ausHwOptions_0_->Y:$6C004,0,16
SI_ausHwOptions_1_->X:$6C004,0,16
SI_ausHwOptions_2_->Y:$6C005,0,16
SI_ausHwOptions_3_->X:$6C005,0,16
SI_usManufacturer->Y:$6C006,0,16
SI_usProductionDate->X:$6C006,0,16
SI_ulLicenseFlags1->DP:$6C007
SI_ulLicenseFlags2->DP:$6C008
SI_usNetxLicenseID->Y:$6C009,0,16
SI_usNetxLicenseFlags->X:$6C009,0,16
SI_usDeviceClass->Y:$6C00A,0,16
SI_bHwRevision->X:$6C00A,0,8
SI_bHwCompatibility->X:$6C00A,8,8
SI_bDevIdNumber->Y:$6C00B,0,8
SCI_bChannelType->Y:$6C00C,0,8
SCI_bSizePositionOfHandshake->X:$6C00C,0,8
SCI_bNumberOfBlocks->X:$6C00C,8,8
SCI_ulSizeOfChannel->DP:$6C00D
SCI_usSizeOfMailbox->Y:$6C00E,0,16
SCI_usMailboxStartOffset->X:$6C00E,0,16
HCI_bChannelType->Y:$6C010,0,8
HCI_ulSizeOfChannel->DP:$6C011
CC0I_bChannelType->Y:$6C014,0,8
CC0I_bChannelId->Y:$6C014,8,8
CC0I_bSizePositionOfHandshake->X:$6C014,0,8
CC0I_bNumberOfBlocks->X:$6C014,8,8
CC0I_ulSizeOfChannel->DP:$6C015
CC0I_usCommunicationClass->Y:$6C016,0,16
CC0I_usProtocolClass->X:$6C016,0,16
CC0I_usConformanceClass->Y:$6C017,0,16
CC1I_bChannelType->Y:$6C018,0,8
CC1I_bChannelId->Y:$6C018,8,8
CC1I_bSizePositionOfHandshake->X:$6C018,0,8
```

```
CC1I_bNumberOfBlocks->X:$6C018,8,8
CC1I_ulSizeOfChannel->DP:$6C019
CC1I_usCommunicationClass->Y:$6C01A,0,16
CC1I_usProtocolClass->X:$6C01A,0,16
CC1I_usConformanceClass->Y:$6C01B,0,16
CC2I_bChannelType->Y:$6C01C,0,8
CC2I_bChannelId->Y:$6C01C,8,8
CC2I_bSizePositionOfHandshake->X:$6C01C,0,8
CC2I_bNumberOfBlocks->X:$6C01C,8,8
CC2I_ulSizeOfChannel->DP:$6C01D
CC2I_usCommunicationClass->Y:$6C01E,0,16
CC2I_usProtocolClass->X:$6C01E,0,16
CC2I_usConformanceClass->Y:$6C01F,0,16
CC3I_bChannelType->Y:$6C020,0,8
CC3I_bChannelId->Y:$6C020,8,8
CC3I_bSizePositionOfHandshake->X:$6C020,0,8
CC3I_bNumberOfBlocks->X:$6C020,8,8
CC3I_ulSizeOfChannel->DP:$6C021
CC3I_usCommunicationClass->Y:$6C022,0,16
CC3I_usProtocolClass->X:$6C022,0,16
CC3I_usConformanceClass->Y:$6C023,0,16
AC0I_bChannelType->Y:$6C024,0,8
AC0I_bChannelId->Y:$6C024,8,8
AC0I_bSizePositionOfHandshake->X:$6C024,0,8
AC0I_bNumberOfBlocks->X:$6C024,8,8
AC0I_ulSizeOfChannel->DP:$6C025
AC1I_bChannelType->Y:$6C028,0,8
AC1I_bChannelId->Y:$6C028,8,8
AC1I_bSizePositionOfHandshake->X:$6C028,0,8
AC1I_bNumberOfBlocks->X:$6C028,8,8
AC1I_ulSizeOfChannel->DP:$6C029
SCtrl_ulSystemCommandCOS->DP:$6C02E
SStat_ulSystemCOS->DP:$6C030
SStat_ulSystemStatus->DP:$6C031
SStat_ulSystemError->DP:$6C032
SStat_ulBootError->DP:$6C033
SStat_ulTimeSinceStart->DP:$6C034
SStat_usCpuLoad->Y:$6C035,0,16
SStat_ulHWFeatures->DP:$6C036
SSMB_usPackagesAccepted->Y:$6C040,0,16
SSMB_ulDest->DP:$6C041
SSMB_ulSrc->DP:$6C042
SSMB_ulDestId->DP:$6C043
SSMB_ulSrcId->DP:$6C044
SSMB_ulLen->DP:$6C045
SSMB_ulId->DP:$6C046
SSMB_ulState->DP:$6C047
SSMB_ulCmd->DP:$6C048
SSMB_ulExt->DP:$6C049
SSMB_ulRout->DP:$6C04A
SSMB_ultData0->DP:$6C04B
SSMB_ultData1->DP:$6C04C
SSMB_ultData2->DP:$6C04D
SSMB_ultData3->DP:$6C04E
SSMB_ultData4->DP:$6C04F
SSMB_ultData5->DP:$6C050
SSMB_ultData6->DP:$6C051
SSMB_ultData7->DP:$6C052
SSMB_ultData8->DP:$6C053
SSMB_ultData9->DP:$6C054
SSMB_ultData10->DP:$6C055
SSMB_ultData11->DP:$6C056
SSMB_ultData12->DP:$6C057
SSMB_ultData13->DP:$6C058
SSMB_ultData14->DP:$6C059
SSMB_ultData15->DP:$6C05A
SSMB_ultData16->DP:$6C05B
SSMB_ultData17->DP:$6C05C
SSMB_ultData18->DP:$6C05D
SSMB_ultData19->DP:$6C05E
SSMB_ultData20->DP:$6C05F
```

```
SRMB_usWaitingPackages->Y:$6C060,0,16
SRMB_ulDest->DP:$6C061
SRMB_ulSrc->DP:$6C062
SRMB_ulDestId->DP:$6C063
SRMB_ulSrcId->DP:$6C064
SRMB_ulLen->DP:$6C065
SRMB_ulId->DP:$6C066
SRMB_ulState->DP:$6C067
SRMB_ulCmd->DP:$6C068
SRMB_ulExt->DP:$6C069
SRMB_ulRout->DP:$6C06A
SRMB_ultData0->DP:$6C06B
SRMB_ultData1->DP:$6C06C
SRMB_ultData2->DP:$6C06D
SRMB_ultData3->DP:$6C06E
SRMB_ultData4->DP:$6C06F
SRMB_ultData5->DP:$6C070
SRMB_ultData6->DP:$6C071
SRMB_ultData7->DP:$6C072
SRMB_ultData8->DP:$6C073
SRMB_ultData9->DP:$6C074
SRMB_ultData10->DP:$6C075
SRMB_ultData11->DP:$6C076
SRMB_ultData12->DP:$6C077
SRMB_ultData13->DP:$6C078
SRMB_ultData14->DP:$6C079
SRMB_ultData15->DP:$6C07A
SRMB_ultData16->DP:$6C07B
SRMB_ultData17->DP:$6C07C
SRMB_ultData18->DP:$6C07D
SRMB_ultData19->DP:$6C07E
SRMB_ultData20->DP:$6C07F
HCSC_bNetxFlags->X:$6C080,0,8
HCSC_NSF_READY->X:$6C080,0,1
HCSC_NSF_ERROR->X:$6C080,1,1
HCSC_NSF_HOST_COS_ACK->X:$6C080,2,1
HCSC_NSF_NETX_COS_CMD->X:$6C080,3,1
HCSC_NSF_SEND_MBX_ACK->X:$6C080,4,1
HCSC_NSF_RECV_MBX_CMD->X:$6C080,5,1
HCSC_bHostFlags->X:$6C080,8,8
HCSC_HSF_RESET->X:$6C080,8,1
HCSC_HSF_BOOTSTART->X:$6C080,9,1
HCSC_HSF_HOST_COS_CMD->X:$6C080,10,1
HCSC_HSF_NETX_COS_ACK->X:$6C080,11,1
HCSC_HSF_SEND_MBX_CMD->X:$6C080,12,1
HCSC_HSF_RECV_MBX_ACK->X:$6C080,13,1
HCCC0_usNetxFlags->Y:$6C082,0,16
HCCC0_NCF_COMMUNICATING->Y:$6C082,0,1
HCCC0_NCF_ERROR->Y:$6C082,1,1
HCCC0_NCF_HOST_COS_ACK->Y:$6C082,2,1
HCCC0_NCF_NETX_COS_CMD->Y:$6C082,3,1
HCCC0_NCF_SEND_MBX_ACK->Y:$6C082,4,1
HCCC0_NCF_RECV_MBX_CMD->Y:$6C082,5,1
HCCC0_NCF_PD0_OUT_ACK->Y:$6C082,6,1
HCCC0_NCF_PD0_IN_CMD->Y:$6C082,7,1
HCCC0_NCF_PD1_OUT_ACK->Y:$6C082,8,1
HCCC0_NCF_PD1_IN_CMD->Y:$6C082,9,1
HCCC0_usHostFlags->X:$6C082,0,16
HCCC0_HCF_HOST_COS_CMD->X:$6C082,2,1
HCCC0_HCF_NETX_COS_ACK->X:$6C082,3,1
HCCC0_HCF_SEND_MBX_CMD->X:$6C082,4,1
HCCC0_HCF_RECV_MBX_ACK->X:$6C082,5,1
HCCC0_HCF_PD0_OUT_CMD->X:$6C082,6,1
HCCC0_HCF_PD0_IN_ACK->X:$6C082,7,1
HCCC0_HCF_PD1_OUT_CMD->X:$6C082,8,1
HCCC0_HCF_PD1_IN_ACK->X:$6C082,9,1
HCCC1_usNetxFlags->Y:$6C083,0,16
HCCC1_NCF_COMMUNICATING->Y:$6C083,0,1
HCCC1_NCF_ERROR->Y:$6C083,1,1
HCCC1_NCF_HOST_COS_ACK->Y:$6C083,2,1
HCCC1_NCF_NETX_COS_CMD->Y:$6C083,3,1
```

```
HCCC1_NCF_SEND_MBX_ACK->Y:$6C083,4,1
HCCC1_NCF_RECV_MBX_CMD->Y:$6C083,5,1
HCCC1_NCF_PD0_OUT_ACK->Y:$6C083,6,1
HCCC1_NCF_PD0_IN_CMD->Y:$6C083,7,1
HCCC1_NCF_PD1_OUT_ACK->Y:$6C083,8,1
HCCC1_NCF_PD1_IN_CMD->Y:$6C083,9,1
HCCC1_usHostFlags->X:$6C083,0,16
HCCC1_HCF_HOST_COS_CMD->X:$6C083,2,1
HCCC1_HCF_NETX_COS_ACK->X:$6C083,3,1
HCCC1_HCF_SEND_MBX_CMD->X:$6C083,4,1
HCCC1_HCF_RECV_MBX_ACK->X:$6C083,5,1
HCCC1_HCF_PD0_OUT_CMD->X:$6C083,6,1
HCCC1_HCF_PD0_IN_ACK->X:$6C083,7,1
HCCC1_HCF_PD1_OUT_CMD->X:$6C083,8,1
HCCC1_HCF_PD1_IN_ACK->X:$6C083,9,1
HCCC2_usNetxFlags->Y:$6C084,0,16
HCCC2_NCF_COMMUNICATING->Y:$6C084,0,1
HCCC2_NCF_ERROR->Y:$6C084,1,1
HCCC2_NCF_HOST_COS_ACK->Y:$6C084,2,1
HCCC2_NCF_NETX_COS_CMD->Y:$6C084,3,1
HCCC2_NCF_SEND_MBX_ACK->Y:$6C084,4,1
HCCC2_NCF_RECV_MBX_CMD->Y:$6C084,5,1
HCCC2_NCF_PD0_OUT_ACK->Y:$6C084,6,1
HCCC2_NCF_PD0_IN_CMD->Y:$6C084,7,1
HCCC2_NCF_PD1_OUT_ACK->Y:$6C084,8,1
HCCC2_NCF_PD1_IN_CMD->Y:$6C084,9,1
HCCC2_usHostFlags->X:$6C084,0,16
HCCC2_HCF_HOST_COS_CMD->X:$6C084,2,1
HCCC2_HCF_NETX_COS_ACK->X:$6C084,3,1
HCCC2_HCF_SEND_MBX_CMD->X:$6C084,4,1
HCCC2_HCF_RECV_MBX_ACK->X:$6C084,5,1
HCCC2_HCF_PD0_OUT_CMD->X:$6C084,6,1
HCCC2_HCF_PD0_IN_ACK->X:$6C084,7,1
HCCC2_HCF_PD1_OUT_CMD->X:$6C084,8,1
HCCC2_HCF_PD1_IN_ACK->X:$6C084,9,1
HCCC3_usNetxFlags->Y:$6C085,0,16
HCCC3_NCF_COMMUNICATING->Y:$6C085,0,1
HCCC3_NCF_ERROR->Y:$6C085,1,1
HCCC3_NCF_HOST_COS_ACK->Y:$6C085,2,1
HCCC3_NCF_NETX_COS_CMD->Y:$6C085,3,1
HCCC3_NCF_SEND_MBX_ACK->Y:$6C085,4,1
HCCC3_NCF_RECV_MBX_CMD->Y:$6C085,5,1
HCCC3_NCF_PD0_OUT_ACK->Y:$6C085,6,1
HCCC3_NCF_PD0_IN_CMD->Y:$6C085,7,1
HCCC3_NCF_PD1_OUT_ACK->Y:$6C085,8,1
HCCC3_NCF_PD1_IN_CMD->Y:$6C085,9,1
HCCC3_usHostFlags->X:$6C085,0,16
HCCC3_HCF_HOST_COS_CMD->X:$6C085,2,1
HCCC3_HCF_NETX_COS_ACK->X:$6C085,3,1
HCCC3_HCF_SEND_MBX_CMD->X:$6C085,4,1
HCCC3_HCF_RECV_MBX_ACK->X:$6C085,5,1
HCCC3_HCF_PD0_OUT_CMD->X:$6C085,6,1
HCCC3_HCF_PD0_IN_ACK->X:$6C085,7,1
HCCC3_HCF_PD1_OUT_CMD->X:$6C085,8,1
HCCC3_HCF_PD1_IN_ACK->X:$6C085,9,1
HCAC0_usNetxFlags->Y:$6C086,0,16
HCAC0_NCF_COMMUNICATING->Y:$6C086,0,1
HCAC0_NCF_ERROR->Y:$6C086,1,1
HCAC0_NCF_HOST_COS_ACK->Y:$6C086,2,1
HCAC0_NCF_NETX_COS_CMD->Y:$6C086,3,1
HCAC0_NCF_SEND_MBX_ACK->Y:$6C086,4,1
HCAC0_NCF_RECV_MBX_CMD->Y:$6C086,5,1
HCAC0_NCF_PD0_OUT_ACK->Y:$6C086,6,1
HCAC0_NCF_PD0_IN_CMD->Y:$6C086,7,1
HCAC0_NCF_PD1_OUT_ACK->Y:$6C086,8,1
HCAC0_NCF_PD1_IN_CMD->Y:$6C086,9,1
HCAC0_usHostFlags->X:$6C086,0,16
HCAC0_HCF_HOST_COS_CMD->X:$6C086,2,1
HCAC0_HCF_NETX_COS_ACK->X:$6C086,3,1
HCAC0_HCF_SEND_MBX_CMD->X:$6C086,4,1
HCAC0_HCF_RECV_MBX_ACK->X:$6C086,5,1
```

```
HCAC0_HCF_PD0_OUT_CMD->X:$6C086,6,1
HCAC0_HCF_PD0_IN_ACK->X:$6C086,7,1
HCAC0_HCF_PD1_OUT_CMD->X:$6C086,8,1
HCAC0_HCF_PD1_IN_ACK->X:$6C086,9,1
HCAC1_usNetxFlags->Y:$6C087,0,16
HCAC1_NCF_COMMUNICATING->Y:$6C087,0,1
HCAC1_NCF_ERROR->Y:$6C087,1,1
HCAC1_NCF_HOST_COS_ACK->Y:$6C087,2,1
HCAC1_NCF_NETX_COS_CMD->Y:$6C087,3,1
HCAC1_NCF_SEND_MBX_ACK->Y:$6C087,4,1
HCAC1_NCF_RECV_MBX_CMD->Y:$6C087,5,1
HCAC1_NCF_PD0_OUT_ACK->Y:$6C087,6,1
HCAC1_NCF_PD0_IN_CMD->Y:$6C087,7,1
HCAC1_NCF_PD1_OUT_ACK->Y:$6C087,8,1
HCAC1_NCF_PD1_IN_CMD->Y:$6C087,9,1
HCAC1_usHostFlags->X:$6C087,0,16
HCAC1_HCF_HOST_COS_CMD->X:$6C087,2,1
HCAC1_HCF_NETX_COS_ACK->X:$6C087,3,1
HCAC1_HCF_SEND_MBX_CMD->X:$6C087,4,1
HCAC1_HCF_RECV_MBX_ACK->X:$6C087,5,1
HCAC1_HCF_PD0_OUT_CMD->X:$6C087,6,1
HCAC1_HCF_PD0_IN_ACK->X:$6C087,7,1
HCAC1_HCF_PD1_OUT_CMD->X:$6C087,8,1
HCAC1_HCF_PD1_IN_ACK->X:$6C087,9,1
CC0_RCX_APP_COS_APP_READY->Y:$6C0C2,0,1
CC0_RCX_APP_COS_BUS_ON->Y:$6C0C2,1,1
CC0_RCX_APP_COS_BUS_ON_ENABLE->Y:$6C0C2,2,1
CC0_RCX_APP_COS_INIT->Y:$6C0C2,3,1
CC0_RCX_APP_COS_INIT_ENABLE->Y:$6C0C2,4,1
CC0_RCX_APP_COS_LOCK_CFG->Y:$6C0C2,5,1
CC0_RCX_APP_COS_LOCK_CFG_ENA->Y:$6C0C2,6,1
CC0_RCX_APP_COS_DMA->Y:$6C0C2,7,1
CC0_RCX_APP_COS_DMA_ENABLE->Y:$6C0C2,8,1
CC0_ulDeviceWatchdog->DP:$6C0C3
CC0_RCX_COMM_COS_READY->Y:$6C0C4,0,1
CC0_RCX_COMM_COS_RUN->Y:$6C0C4,1,1
CC0_RCX_COMM_COS_BUS_ON->Y:$6C0C4,2,1
CC0_RCX_COMM_COS_CONFIG_LOCKED->Y:$6C0C4,3,1
CC0_RCX_COMM_COS_CONFIG_NEW->Y:$6C0C4,4,1
CC0_RCX_COMM_COS_RESTART_REQ->Y:$6C0C4,5,1
CC0_RCX_COMM_COS_RESTART_REQ_ENA->Y:$6C0C4,6,1
CC0_RCX_COMM_COS_DMA->Y:$6C0C4,7,1
CC0_ulCommunicationState->DP:$6C0C5
CC0_ulCommunicationError->DP:$6C0C6
CC0_usVersion->Y:$6C0C7,0,16
CC0_usWatchdogTime->X:$6C0C7,0,16
CC0_bPDInHskMode->Y:$6C0C8,0,8
CC0_bPDInSource->Y:$6C0C8,8,8
CC0_bPDOutHskMode->X:$6C0C8,0,8
CC0_bPDOutSource->X:$6C0C8,8,8
CC0_ulHostWatchdog->DP:$6C0C9
CC0_ulErrorCount->DP:$6C0CA
CC0_bErrorLogInd->Y:$6C0CB,0,8
CC0_bErrorPDInCnt->Y:$6C0CB,8,8
CC0_bErrorPDOutCnt->X:$6C0CB,0,8
CC0_bErrorSyncCnt->X:$6C0CB,8,8
CC0_bSyncHskMode->Y:$6C0CC,0,8
CC0_bSyncSource->Y:$6C0CC,8,8
CC0_ulSlaveState->DP:$6C0CE
CC0_ulSlaveErrLogInd->DP:$6C0CF
CC0_ulNumOfConfigSlaves->DP:$6C0D0
CC0_ulNumOfActiveSlaves->DP:$6C0D1
CC0_ulNumOfDiagSlaves->DP:$6C0D2
CC1_RCX_APP_COS_APP_READY->Y:$6D002,0,1
CC1_RCX_APP_COS_BUS_ON->Y:$6D002,1,1
CC1_RCX_APP_COS_BUS_ON_ENABLE->Y:$6D002,2,1
CC1_RCX_APP_COS_INIT->Y:$6D002,3,1
CC1_RCX_APP_COS_INIT_ENABLE->Y:$6D002,4,1
CC1_RCX_APP_COS_LOCK_CFG->Y:$6D002,5,1
CC1_RCX_APP_COS_LOCK_CFG_ENA->Y:$6D002,6,1
CC1_RCX_APP_COS_DMA->Y:$6D002,7,1
```

```
CC1_RCX_APP_COS_DMA_ENABLE->Y:$6D002,8,1
CC1_ulDeviceWatchdog->DP:$6D003
CC1_RCX_COMM_COS_READY->Y:$6D004,0,1
CC1_RCX_COMM_COS_RUN->Y:$6D004,1,1
CC1_RCX_COMM_COS_BUS_ON->Y:$6D004,2,1
CC1_RCX_COMM_COS_CONFIG_LOCKED->Y:$6D004,3,1
CC1_RCX_COMM_COS_CONFIG_NEW->Y:$6D004,4,1
CC1_RCX_COMM_COS_RESTART_REQ->Y:$6D004,5,1
CC1_RCX_COMM_COS_RESTART_REQ_ENA->Y:$6D004,6,1
CC1_RCX_COMM_COS_DMA->Y:$6D004,7,1
CC1_ulCommunicationState->DP:$6D005
CC1_ulCommunicationError->DP:$6D006
CC1_usVersion->Y:$6D007,0,16
CC1_usWatchdogTime->X:$6D007,0,16
CC1_bPDInHskMode->Y:$6D008,0,8
CC1_bPDInSource->Y:$6D008,8,8
CC1_bPDOutHskMode->X:$6D008,0,8
CC1_bPDOutSource->X:$6D008,8,8
CC1_ulHostWatchdog->DP:$6D009
CC1_ulErrorCount->DP:$6D00A
CC1_bErrorLogInd->Y:$6D00B,0,8
CC1_bErrorPDInCnt->Y:$6D00B,8,8
CC1_bErrorPDOutCnt->X:$6D00B,0,8
CC1_bErrorSyncCnt->X:$6D00B,8,8
CC1_bSyncHskMode->Y:$6D00C,0,8
CC1_bSyncSource->Y:$6D00C,8,8
CC1_ulSlaveState->DP:$6D00E
CC1_ulSlaveErrLogInd->DP:$6D00F
CC1_ulNumOfConfigSlaves->DP:$6D010
CC1_ulNumOfActiveSlaves->DP:$6D011
CC1_ulNumOfDiagSlaves->DP:$6D012
```

DPRAM DATA PROCESSING

Since there are two processors (i.e. UMAC and netX) attempting to access data registers in Dual-Ported Memory simultaneously, several handshaking modes can be used to guarantee data consistency. Each sub-block defines the type of handshaking, if any, it requires. The ACC-72EX Setup Assistant software output file lists the type of handshaking required for each of the sub-blocks available on the COMX module.

Should handshaking not be used, collision circuitry on the gateway will, in the very least, guarantee consistency within single byte boundaries.

Non-Cyclic Data Exchange

The mailbox of a communication channel or system channel has two areas that are used for non-cyclic message transfer to and from the netX.

- Send Mailbox (System / Communication Channel)

  Packet transfer from UMAC to netX firmware

- Receive Mailbox (System / Communication Channel)

  Packet transfer from netX firmware to UMAC

For a communication channel, send and receive mailbox areas are used by fieldbus protocols, providing a non-cyclic data exchange mechanism. Another use of the mailbox system is to allow access to the firmware running on the netX chip for diagnostic and identification purposes. The **send mailbox** is used to transfer cyclic data **to** the network or **to** the netX. The **receive mailbox** is used to transfer cyclic data **from** the network or **from** the netX. Modbus Plus or Ethernet TCP/IP is an example of a fieldbus protocol which utilizes a non-cyclic data exchange.

Whether or not a mailbox is used depends on the function of the firmware.

Note

Each mailbox can hold one packet at a time. netX stores packets in an internal packet queue; these packets are not retrieved by UMAC. This queue has limited space and may fill up, so new packets may be lost. To avoid these deadlock situations, it is strongly recommended to empty the mailbox frequently, even if packets are not expected by the UMAC program. Unexpected command packets should be returned to the sender with an Unknown Command in the status field; unexpected reply messages can be discarded.

Message or Packets

The non-cyclic packets obtained through the netX mailbox have the following structure:

	Hilscher Documentation	ACC-72EX Setup Assistant
System Block Send Mailbox	usPackagesAccepted	SSMB_usPackagesAccepted
	ulDest	SSMB_ulDest
	ulSrc	SSMB_ulSrc
	ulDestId	SSMB_ulDestId
	ulSrcId	SSMB_ulSrcId
	ulLen	SSMB_ulLen
	ulId	SSMB_ulId
	ulState	SSMB_ulState
	ulCmd	SSMB_ulCmd
	ulExt	SSMB_ulExt
	ulRout	SSMB_ulRout
	…	SSMB_ultData0 .. SSMB_ultData20

	Hilscher Documentation	ACC-72EX Setup Assistant
System Block Receive Mailbox	usWaitingPackages	SRMB_usWaitingPackages
	ulDest	SRMB_ulDest
	ulSrc	SRMB_ulSrc
	ulDestId	SRMB_ulDestId
	ulSrcId	SRMB_ulSrcId
	ulLen	SRMB_ulLen
	ulId	SRMB_ulId
	ulState	SRMB_ulState
	ulCmd	SRMB_ulCmd
	ulExt	SRMB_ulExt
	ulRout	SRMB_ulRout
	…	SRMB_ultData0 .. SRMB_ultData20

The size of a packet is always at least 40 bytes. Depending on the command, a packet may or may not have a payload in the data field (tData). If present, the contents of the data field are specific to the command or reply.

- Destination Queue Handler

  The ulDest field identifies a task queue in the context of the netX firmware. The task queue represents the final receiver of the packet and is assigned to a protocol stack. The ulDest field has to be filled out in any case; otherwise, the netX operating system cannot route the packet.

- Source Queue Handler

  The ulSrc field identifies the sender of the packet. In the context of the netX firmware (inter-task communication), this field holds the identifier of the sending task. Usually, a UMAC program uses this field for its own handle, but it can hold any handle of the sending process. The receiving task does not evaluate this field and will pass it back unchanged to the originator of the packet.

- Destination Identifier

  The ulDestId field identifies the destination of an unsolicited packet from the netX firmware to the UMAC. It can hold any handle that helps identify the receiver. Its use is mandatory for unsolicited packets. The receiver of unsolicited packets has to register for this service (details are yet to be determined).

- Source Identifier

  The ulSrcId field identifies the originator of a packet. This field is used by a UMAC program which passes a packet from an external process to an internal netX task. The ulSrcId field holds the handle of the external process. When the netX operating system returns the packet, the UMAC program can identify the packet, and returns it to the originating process. The receiving task on the netX does not evaluate this field, and passes it back unchanged. For inter-task communication, this field is not used.

- Length of Data Field

  The ulLen field holds the size of the data field tData in bytes. It defines the total size of the packet's payload that follows the packet's header. Note that the size of the header is not included in ulLen. Depending on the command or reply, a data field may or may not be present in a packet. If no data field is used, the length field is set to zero.

- Identifier

  The ulId field is used to identify a specific packet among others of the same kind. That way the application or driver can match a specific reply or confirmation packet to a previous request packet. The receiving task does not change this field and passes it back to the originator of the packet. Its use is optional in most of cases, but it is mandatory for fragmented packets. Example: downloading large amounts of data that do not fit into a single packet. For fragmented packets, the identifier field is incremented by one for every new packet.

- Status / Error Code

  The ulSta field is used in response or confirmation packets. It informs the originator of the packet about success or failure of the execution of the command. The field may be also used to hold status information in a request packet. Status and error codes that may be returned in ulSta are outlined in Status and Error Code section.

- Command / Response

  The ulCmd field holds the command code or the response code. The command/response is specific to the receiving task. If a task is not able to execute certain commands, it will return the packet with an error indication. A command is always even (the least significant bit is zero). In the response packet, the command code is incremented by one indicating a confirmation to the request packet.

- Extension

  The extension field ulExt is used for controlling packets that are sent in a sequenced or fragmented manner. The extension field indicates the first, last, or a packet of a sequence. If fragmentation of packets is not required, the extension field is set to zero.

- Routing Information

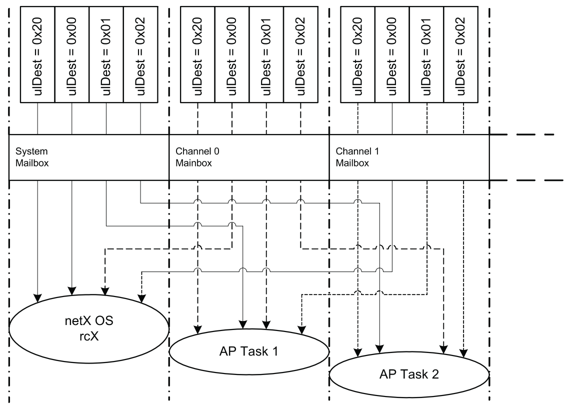
  The ulRout field is used internally by the netX firmware only. It has no meaning to a driver type application and therefore is set to zero.

- User Data Field

  The tData field contains the payload of the packet. Depending on the command or reply, a packet may or may not have a data field. The length of the data field is given in the ulLen field.

About System and Channel Mailbox

The preferred way to address the netX operating system, rcX, is through the system mailbox. The preferred way to address a protocol stack is through its channel mailbox. All mailboxes, however, have a mechanism to route packets to any communication channel or the system channel. Therefore, the destination identifier ulDest in a packet header has to be filled in according to the targeted receiver (see the following image).



The above figure and table below illustrate the use of the destination identifier ulDest.

Value	Definition / Description
$0	Packet is passed to the netX operating system rcX
$1	Packet is passed to communication channel 0
$2	Packet is passed to communication channel 1
$3	Packet is passed to communication channel 2
$4	Packet is passed to communication channel 3
$20	Packet is passed to ‘local’ communication or system channel
Else	Reserved, Do Not Use

In regards to the channel identifier 0x00000020 (= Channel Token), the Channel Token is valid for any mailbox. That way, the UMAC program uses the same identifier for all packets without actually knowing which mailbox or communication channel is applied. The packet stays “local.” The system mailbox is a little bit different because it is used to communicate to the netX operating system, rcX. The rcX has its own range of valid command codes and differs from the communication channels.

If there is a reply packet, the netX operating system returns it to the same mailbox that the request packet went through. Consequently, the UMAC program has to return its reply packet to the mailbox from which the request was received.

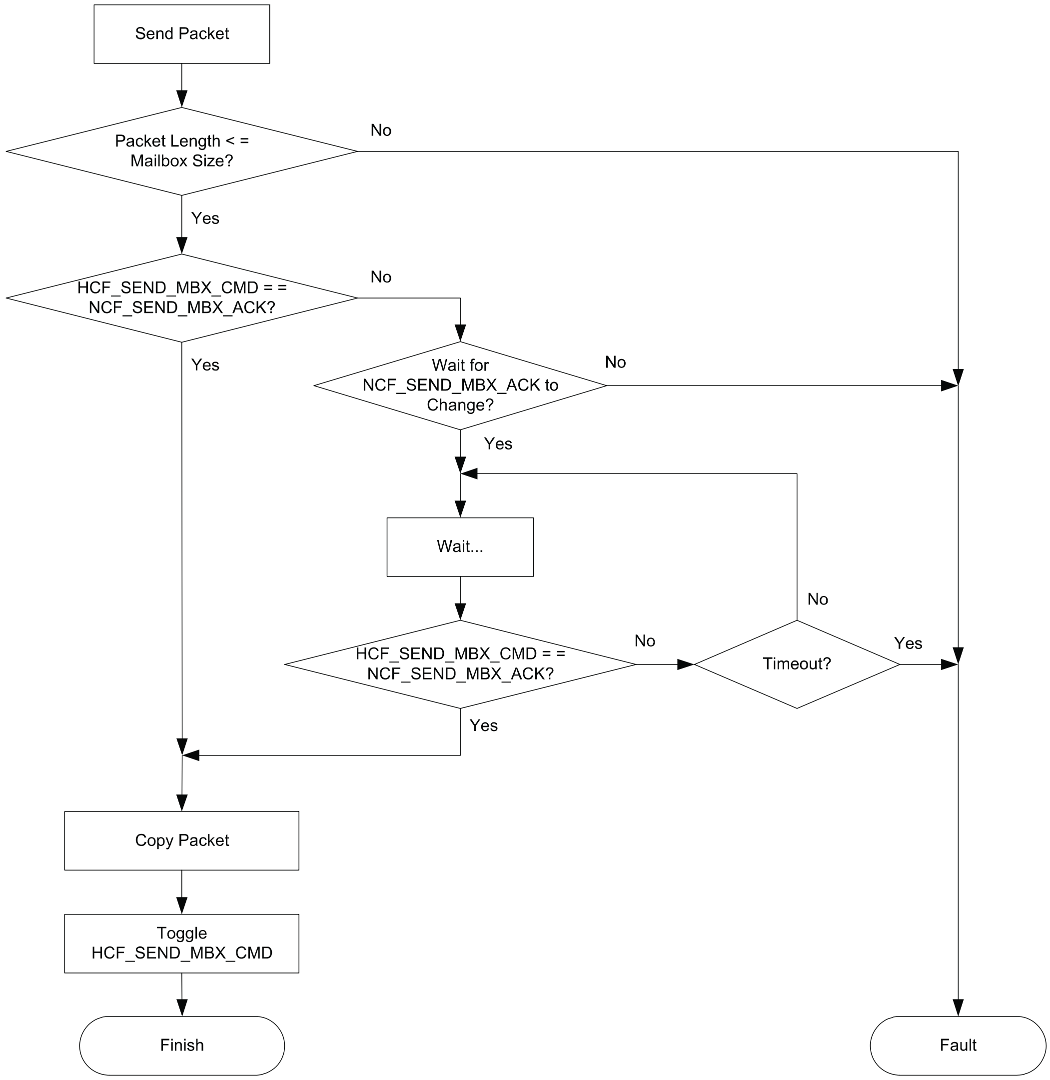
Command and Acknowledge

To ensure data consistency over the content of a mailbox, the firmware uses a pair of flags, each for one direction. Engaging these flags gives access rights alternating to either the user application or the netX firmware. If both UMAC and netX firmware were to access the mailbox at the same time, it may cause loss of data or inconsistency.

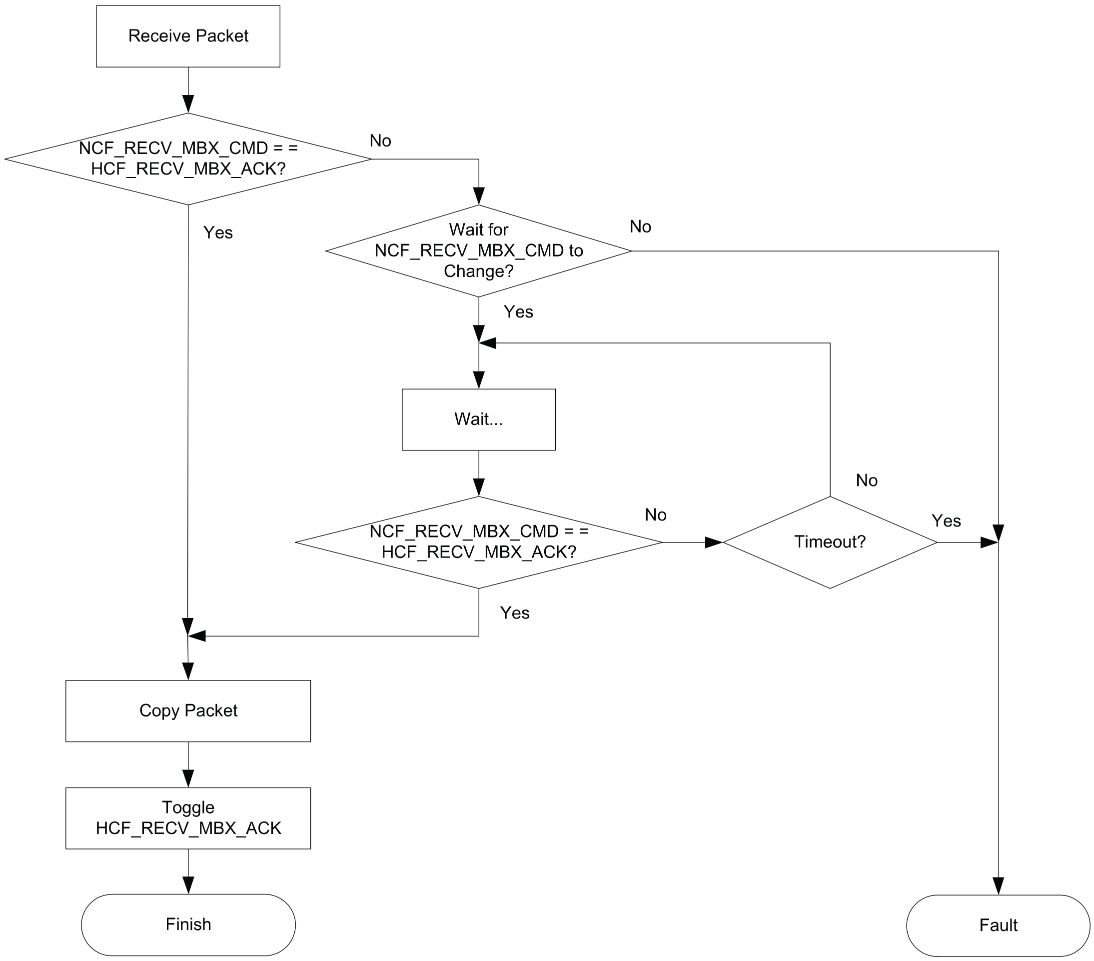
As a general rule, if both flags have the same value (both are set or both are cleared), the process which intends to write has access rights. If they have a different value, the process which intends to read has access rights. The following table illustrates this mechanism.

Send Mailbox	**CMD Flag**	**ACK Flag**	
UMAC Has Write Access	0	0	netX Has NO Read Access
UMAC Has NO Write Access	0	1	netX Has Read Access
UMAC Has NO Write Access	1	0	netX Has Read Access
UMAC Has Write Access	1	1	netX Has NO Read Access
Receive Mailbox	**CMD Flag**	**ACK Flag**	
UMAC Has NO Read Access	0	0	netX Has Write Access
UMAC Has Read Access	0	1	netX Has NO Write Access
UMAC Has Read Access	1	0	netX Has NO Write Access
UMAC Has NO Read Access	1	1	netX Has Write Access

The following flowcharts illustrate how the transfer mechanism (send and receive packets) works. In order to send a packet, first the function checks if the size of the packet to be sent exceeds the mailbox size. If both the Host Send Mailbox Command flag and the netX Send Mailbox Acknowledge flag are either set or cleared, the host application is allowed to send the packet. When copying data to the mailbox is done, the host toggles the Host Send Mailbox Command flag to give control to the netX firmware.

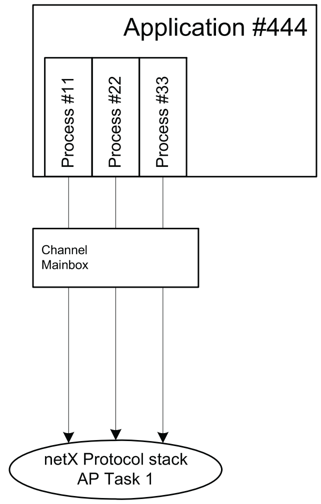


In order to receive a packet, the function checks if the netX Receive Mailbox Command flag and the Host Receive Mailbox Acknowledge flag have different values. If so, the host application is allowed to access the mailbox. When the host is done copying data from the mailbox, the host toggles the Host Receive Mailbox Acknowledge flag to give control to the netX firmware.



Using ulSrc and ulSrcId

Generally, a netX protocol stack is addressed through its communication channel mailbox. The example below shows how a host application addresses a protocol stack running in the context of the netX chip. The application is identified by a number (#444 in this example). The application consists of three processes numbered #11, #22 and #33. These processes communicate through the channel mailbox to the AP task of a protocol stack. See the following image:



Example:

This example applies to command messages imitated by a process in the context of the host application identified by number #444. If the process #22 sends a packet through the channel mailbox to the AP task, the packet header has to be filled in as follows:

Destination Queue Handler	`ulDest   = 32;  /* 0x20: local channel mailbox */`
Source Queue Handler	`ulSrc    = 444; /* host application */`
Destination Identifier	`ulDestId= 0;    /* not used */`
Source Identifier	`ulSrcId = 22;   /* process number */`

For packets through the channel mailbox, the application uses 32 (= 0x20, Channel Token) for the destination queue handler ulDest. The source queue handler ulSrc and the source identifier ulSrcId are used to identify the originator of a packet. The destination identifier ulDestId can be used to address certain resources in the protocol stack. It is not used in this example. The source queue handler ulSrc must have an entry, and therefore its use is mandatory; the use of ulSrcId is optional.

The netX operating system passes the request packet to the protocol stack's AP task. The protocol stack then builds a reply to the packet and returns it to the mailbox. The application has to make sure that the packet finds its way back to the originator (process #22 in the example).

How to Route rcX Packets

To route an rcX packet, the source identifier ulSrcId and the source queues handler ulSrc in the packet header hold the identification of the originating process. The router saves the original handle from ulSrcId and ulSrc. It uses handles of its own choice for ulSrcId and ulSrc before it sends the packet to the receiving process. That way, the router can identify the corresponding reply packet and match the handle from that packet with the one stored earlier. Lastly, the router replaces its handles with the original handles and returns the packet to the originating process.

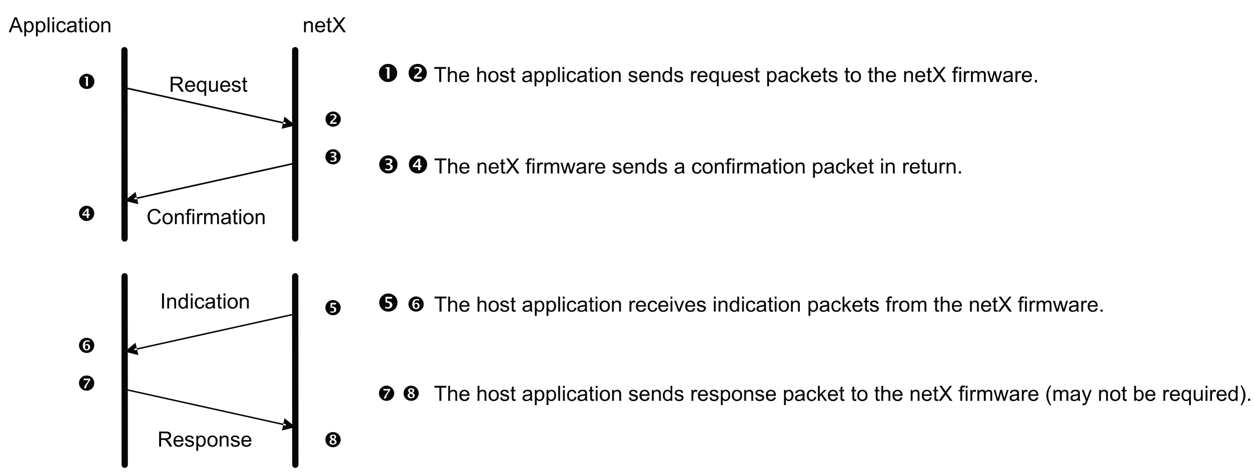
Client/Server Mechanism

Depending on the message destination or packet protocol, the UMAC program or application can act as a client or a server. This section explains both methods, but selection of the appropriate method depends on the destination or protocol option.

Application as Client

The host application may send request packets to the netX firmware at any time (transition 1 ⇨ 2). Depending on the protocol stack running on the netX, parallel packets are not permitted (see protocol specific manual for details). The netX firmware sends a confirmation packet in return, signaling success or failure (transition 3 ⇨ 4) while processing the request.

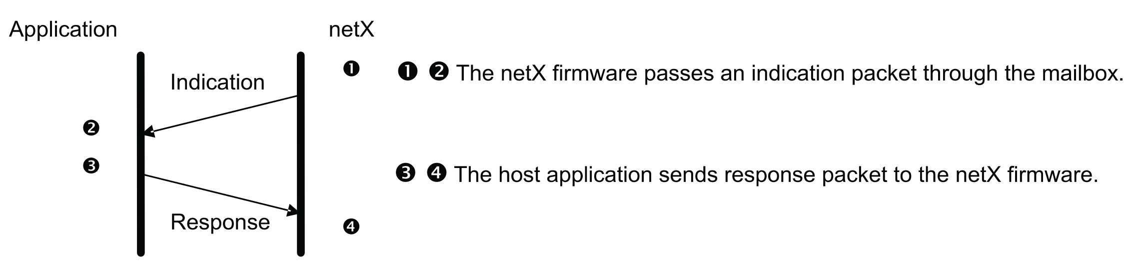
The host application has to register with the netX firmware in order to receive indication packets (transition 5 ⇨ 6). Depending on the protocol stack, this is done either implicitly (if application opens a TCP/UDP socket) or explicitly (if application wants to receive unsolicited DPV1 packets). Details on when and how to register for certain events is described in the protocol specific manual. Depending on the command code of the indication packet, a response packet to the netX firmware may or may not be required (transition 7 ⇨ 8).



Application as Server

The host application has to register with the netX firmware in order to receive indication packets (unsolicited telegrams). Depending on the protocol stack, this is done either implicitly (if the application opens a TCP/UDP socket) or explicitly (if the application wants to receive unsolicited DPV1 packets). Details on when and how to register for certain events is described in the protocol-specific manual.

When an appropriate event occurs and the host application is registered to receive such a notification, the netX firmware passes an indication packet through the mailbox (transition 1 ⇨ 2). The host application is expected to send a response packet back to the netX firmware (transition 3 ⇨ 4).



Input/Output Data Image

Hilscher products support two methods for accessing the Input/Output Data Image:

- DPM (Dual-Ported Memory) Mode
- DMA (Direct Memory Access) Mode

However, the modules used in ACC-72EX only support the DPM mode, and only Hilscher PCI cards support DMA mode.

In DPM Mode, handshaking between the UMAC (host) program and netX is required for any data transfer.

Process Data Handshake Modes

The netX firmware allows controlling the transfer of data independently for inputs and outputs. Therefore, the process data handshake is carried out individually for input and output image. The handshake cells are located in the handshake channel.

Mode	Controlled by	Consistency	Supported by
Buffered	Host (Application/Driver)	Yes	Master & Slave Firmware

Buffered, Host Controlled Mode

The Buffered data transfer mode can be used for both master- and slave- type devices. In "buffered" mode, the protocol stack handles the exchange of data between internal buffers and the process data images in the dual-port memory with the application via a handshake mechanism. Once copied from/into the input/output area, the host application gives control over the dual-port memory to the protocol stack. Control is given back to the host application when the protocol stack has finished copying, and so on.

Note

The network cycle and the task cycle of the host application are not synchronized, but are consistent.

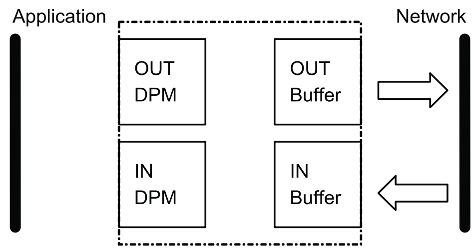
If the host application is faster than the network cycle, it might be possible that data in the output buffers is overwritten without ever being sent to the network. As for the other direction, the host application may read the same input values over several read cycles.

If the host application is slower than the network cycle, the protocol stack overwrites the input buffer with new data received from the network, which were never received by the host application. The output data on the network will be the same over several network cycles.

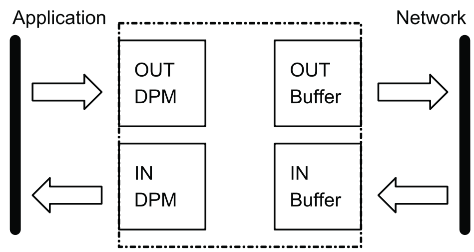
For each valid bus cycle, the protocol stack updates the process data in the internal input buffer. When the application toggles the appropriate input handshake bit, the protocol stack copies the data from the internal IN buffer into the input data image of the dual-port memory. Now the application can copy data from the dual-port memory and then give control back to the protocol stack by toggling the appropriate input handshake bit. When the application/driver toggles the output handshake bit, the protocol stack copies the data from the output data image of the dual-port memory into the internal buffer. From there, the data is transferred to the network. The protocol stack toggles the appropriate handshake bits back, indicating to the application that the transfer is finished and a new data exchange cycle may start. This mode guarantees data consistency over both the input and output areas.

Step-by-Step Procedure

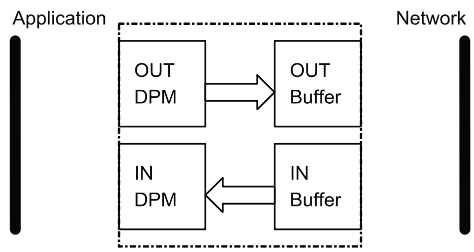
Step 1 The protocol stack sends data from the internal OUT buffer to the network and receives data from the network in the internal IN buffer.



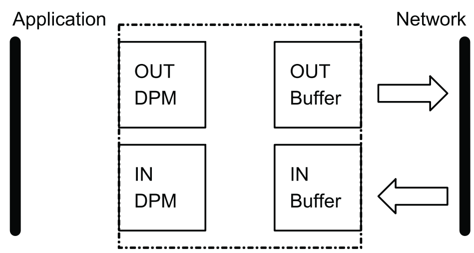
Step 2 The application has control over the dual-port memory and exchanges data with the input and output data images in the dual-port memory. The application then toggles the handshake bits, giving control over the dual-port memory to the protocol stack



Step 3 The protocol stack copies the content of the output data image into the internal OUT buffer, and from the IN buffer to the input data image.



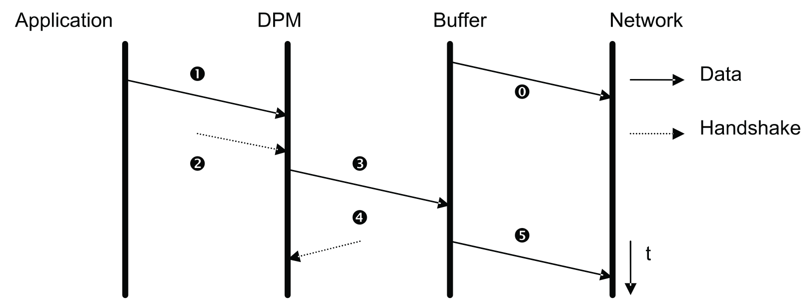
Step 4 The protocol stack toggles the handshake bits, giving control back to the application. Now, the protocol stack uses the new output data image from the OUT buffer to send it to the network, and receives data into the internal IN buffer, and then the cycle repeats.



Time-Related View

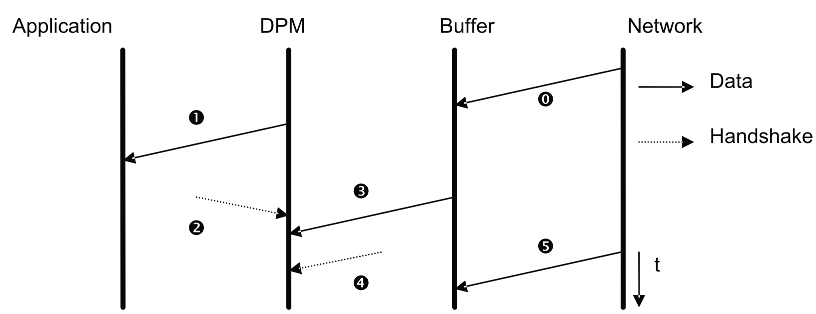
The following figure shows the procedure in a time-related view.

Output Data Exchange



1. The protocol stack constantly transmits data from the buffer to the network.
2. The application has control over the dual-port memory and can copy data to the output data image.
3. The application then toggles the handshake bits, giving control over the dual-port memory to the protocol stack.
4. The protocol stack copies the content of the output data image into the internal OUT buffer.
5. The protocol stack toggles the handshake bits, giving control back to the application.
6. Once updated, the protocol stack uses the new data from the internal buffer and sends it to the network. The cycle repeats with step 1.

Input Data Exchange



1. The protocol stack constantly receives data from the network into the buffer.
2. The application has control over the dual-port memory input data image and exchanges data with the input data image in the dual-port memory.
3. The application then toggles the handshake bits, giving control over the dual-port memory to the netX protocol stack.
4. The protocol stack copies the latest content of the internal IN buffer to the input data image of the dual-port memory.
5. The protocol stack then toggles the handshake bits, giving control back to the application.
1. The protocol stack receives data from the network into the buffer (i.e. the cycle starts over with the first step).

Note

In case of a network fault (e.g. disconnected network cable), a slave firmware keeps the last state of the input data image. As soon as the firmware detects the network fault, it clears the Communicating flag in the netX communication flags. The input data should then no longer be evaluated.

Start / Stop Communication

Controlled or Automatic Start

The firmware has the option to start network communication after power up automatically. Whether or not the network communication will be started automatically is configurable. However, the preferred option is called “Controlled Start of Communication.” This option forces the channel firmware to wait for the host application to allow network connection being opened by setting the Bus On flag in the Application Change of State register in the channel’s control block. Consequently, the protocol stack will not allow the opening of network connections and does not exchange any cyclic process data until the Bus On flag is set.

The second option enables the channel firmware to open network connections automatically without interacting with the host application. It is called “Automatic Start of Communication.” This method is not recommended because the host application has no control over the network connection status. In this case, the Bus On flag is not evaluated.

Note

The Controlled Start of communication is the default method used for the default dual-port memory layout.

Start / Stop Communication through Dual-Port Memory

(Re-)Start Communication

To allow the protocol stack to open connections or to allow connections to be opened, the application sets the Bus On flag in the Application Change of State register in the channel’s control block. When firmware has established a cyclic connection to at least one network mode, the channel firmware sets the Communicating flag in the netX Communication Flags register.

Stop Communication

To force the channel firmware to disable all network connections, the host application clears the Bus On flag in the “Application Change of State” register in the channel’s control block. The firmware then closes all open network connections. A slave protocol stack would reject attempts to reopen a connection until the application allows opening network connections again (Bus On flag is set). When all connections are closed, the channel firmware clears the Communicating flag in the netX Communication Flags register.

Reset Command

System Reset vs. Channel Initialization

There are several methods to restart the netX firmware. The first is called "System Reset." The System Reset affects the netX operating system, rcX, and the protocol stacks. It forces the chip to immediately stop all running protocol stacks and the rcX itself. During the system reset, the netX is performing an internal memory check and other functions to insure the integrity of the netX chip itself.

The Channel Initialization, as the second method, affects a communication channel only. The channel firmware then reads and evaluates the configuration settings (or SYCON.net database, if available) again. The operating system is not affected. There are no particular tests performed during a channel initialization.

A third method to reset the netX chip is called Boot Start. No firmware is started when a System Reset is executed with the boot start flag set. The netX remains in boot loader mode.

Note

A System Reset, Channel Initialization, and boot start may cause all network connection to be interrupted immediately, regardless of their current state.

During a HW-Reset and the time when the 2nd stage loader starts the Firmware, the content of the dual port memory can be 0xFFFF or 0x0BAD for a short period of time.

When used with Turbo PMAC2 CPU, it is necessary to reset the COMX module for proper functionality after initial power up, cycle power, or a $$$ or $$$*** command.

Resetting netX through Dual-Port Memory

To reset the entire netX firmware, the host application has to set the HSF_RESET bit in the bHostSysFlags register to perform a system-wide reset and respectively the APP_COS_INIT flag for a channel initialization in the ulApplicationCOS variable in the control block of the channel. The system reset and the channel initialization are handled differently by the firmware (see above).

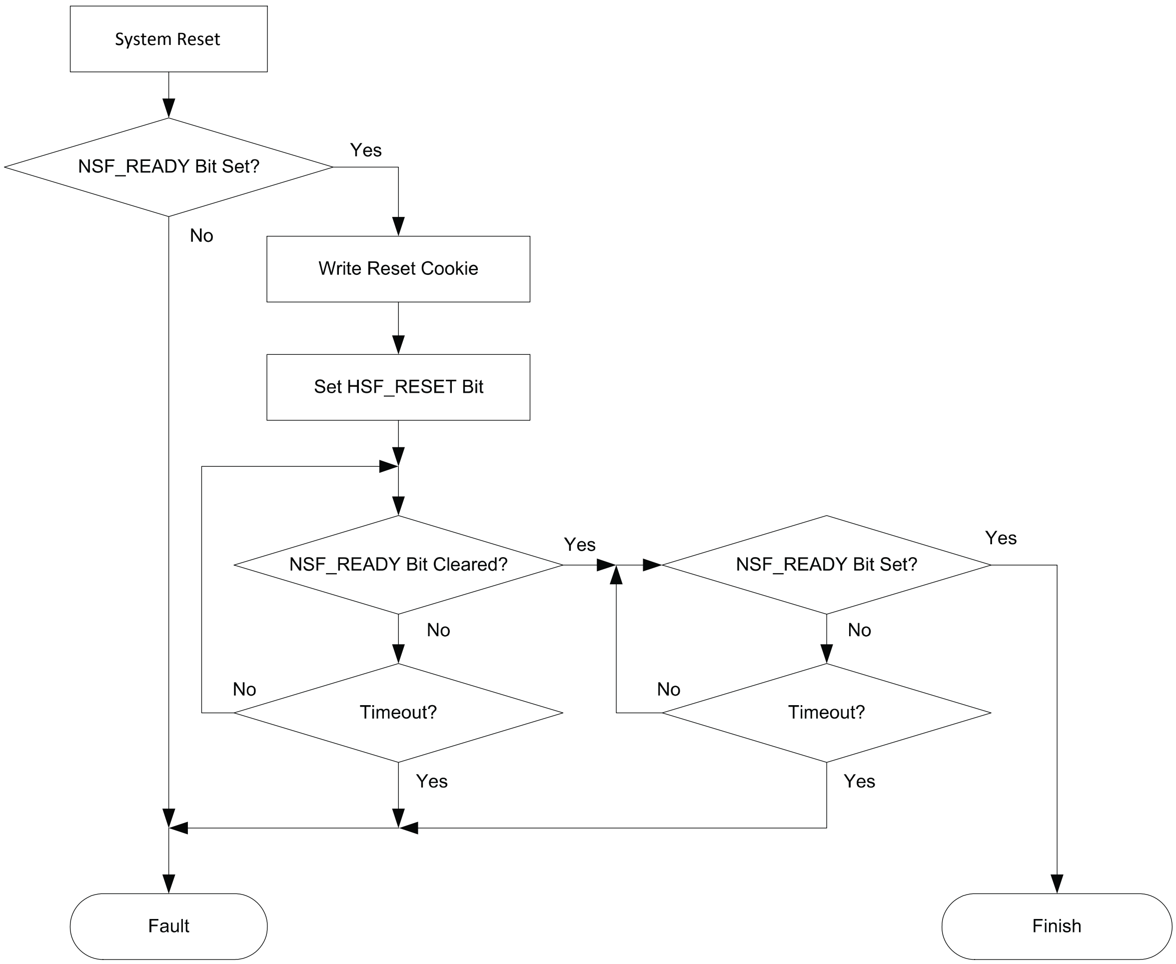
System Reset

To reset the netX operating system rcX and all communication channels, the host application has to write $55AA55AA (System Reset Cookie) to the ulSystemCommandCOS variable in the system control block. Then, the HSF_RESET flag in bHostSysFlags has to be set. If the operating system does not find $55AA55AA in the ulSystemCommandCOS variable, the reset command will be ignored.

The operating system clears the NSF_READY flag in bNetxFlags in the system handshake register, indicating that the system-wide reset is in progress. During the reset, all communication channel tasks are stopped, regardless of their current state. The rcX operating system flushes the entire dual-port memory and writes all memory locations to zero. After the reset, if rcX is finished without complications, and all protocol stacks are started properly, the NSF_READY flag is set again. Otherwise, the NSF_ERROR flag in bNetxFlags in the system handshake register is set, and an error code is written in ulSystemError in the system status block (see page 46), which helps identify possible problems.

Value	Definition/Description
$55AA55AA	System reset cookie

The image below illustrates the steps the host application has to perform in order to execute a systemwide reset on the netX chip through the dual-port memory.



Timing

The duration of the reset outlined above depends on the firmware. Typically, the NSF_READY flag is cleared within around 100 – 500 ms after the HSF_RESET Flag was set. When cleared, the NSF_READY bit will be set again after around 0.5 – 5 s. Generally, the reset should not take more than 6 seconds.

Channel Initialization

In order to force the protocol stack to restart and evaluate the configuration parameter again, the application can set the APP_COS_INIT flag in the ulApplicationCOS register in the control block or send a reset packet to the communication channel. All open network connections are interrupted immediately, regardless of their current state. Reinitializing the channel is not allowed if the database is locked.

Changing flags in the ulApplicationCOS register requires the application also to toggle the host change of state command flag in the host communication flags register. Only then, the netX protocol stack recognizes the reset command.

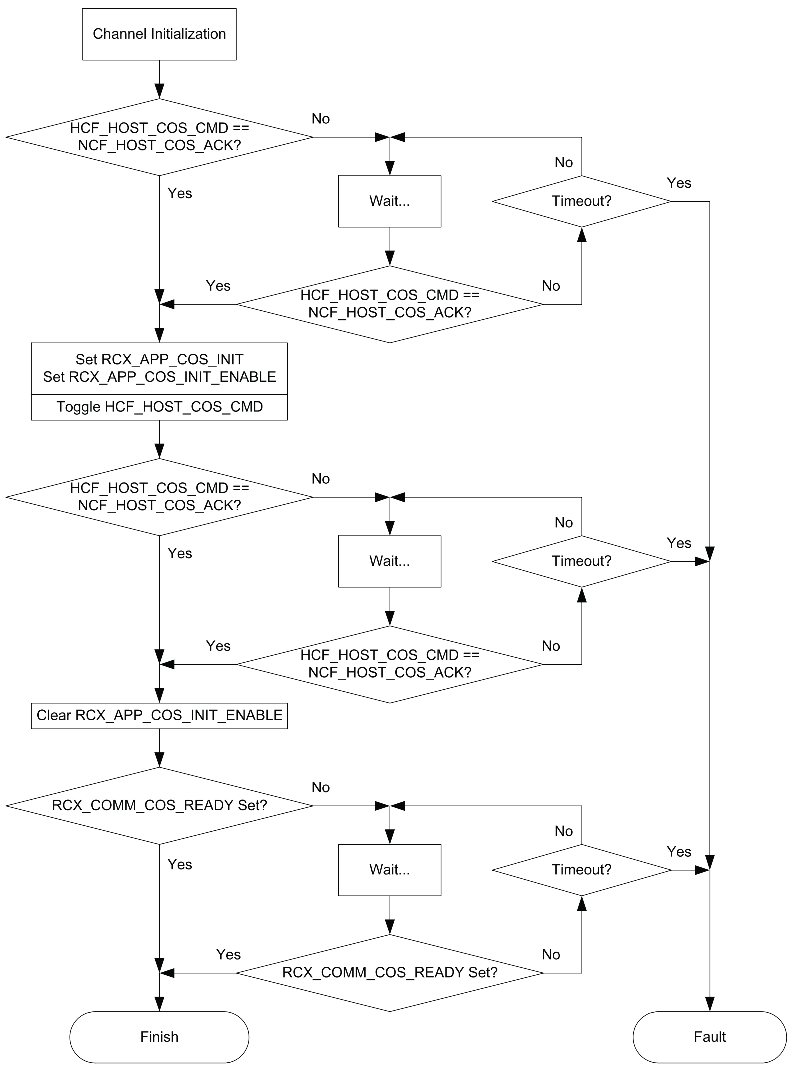
Below is the sequence:

```
CC0_RCX_APP_COS_INIT=1
CC0_RCX_APP_COS_INIT_ENABLE=1
HCCC0_HCF_HOST_COS_CMD=1
```

During channel initialization, the RCX_COMM_COS_READY flag and the RCX_COMM_COS_RUN flag are cleared together. The RCX_COMM_COS_READY flag stays cleared for at least 20 ms before it is set again, indicating that the initialization has finished. The RCX_COMM_COS_RUN flag is set if a valid configuration was found. Otherwise, it stays cleared.

After the initialization process has finished, the protocol stack checks ulApplicationCOS register. If the RCX_APP_COS_BUS_ON flag and the RCX_APP_COS_BUS_ON_ENABLE flags are set, network communication will be restored automatically. The same is true for the Lock Configuration feature (RCX_APP_COS_LOCK_CONFIG / RCX_APP_COS_LOCK_CONFIG_ENABLE) and the DMA data transfer mechanism (RCX_APP_COS_DMA / RCX_APP_COS_DMA_ENABLE).

The image below illustrates the steps the host application has to perform in order to execute a channel initialization on the protocol stack through the dual-port memory.



System Reset through Packets

The netX chip can be reset using a packet instead of the dual-port memory. The request packet is passed through the system mailbox. All open network connections are interrupted immediately, regardless of their current state. Reinitializing the channel is not allowed if the database is locked.

For detailed information about reset message settings, please see Hilscher documentation.

SOFTWARE SETUP

ACC-72EX supports multiple protocols, and setting up each protocol can be a bit different, as described in the protocol specific documentation provided by Hilscher. In this section, most of the generic steps are covered with the help of examples and screenshots.

Required Software Packages

Two software packages are required for setting up ACC-72EX:

1. SYCON.NET (V1.310.x.x or newer), available through Hilscher's website.
2. ACC-72EX Setup Assistant Software.

Both software packages have to be installed on the PC used for initial setup of the system and commissioning of the machine. Notice that neither of these software packages is required after the initial setup and the unit can work as a standalone setup.

SyCon.NET Software Setup

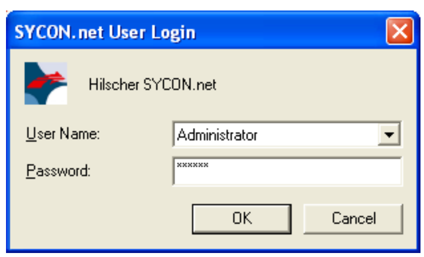
SYCON.net is a tool for the configuration of Fieldbus and Real-Time Ethernet systems. It is based on the standardized FDT / DTM technology. Online diagnostic indicators and auto-scan function for the reading of network participants assist in the commissioning of the network. SYCON.NET is provided with the gateway module under license from Hilscher Corporation.

With the power off, plug the ACC-72EX into the UBUS backplane and turn on the power to the UMAC rack. Connect the diagnostic port to a USB port on the PC using a micro-USB type cable.

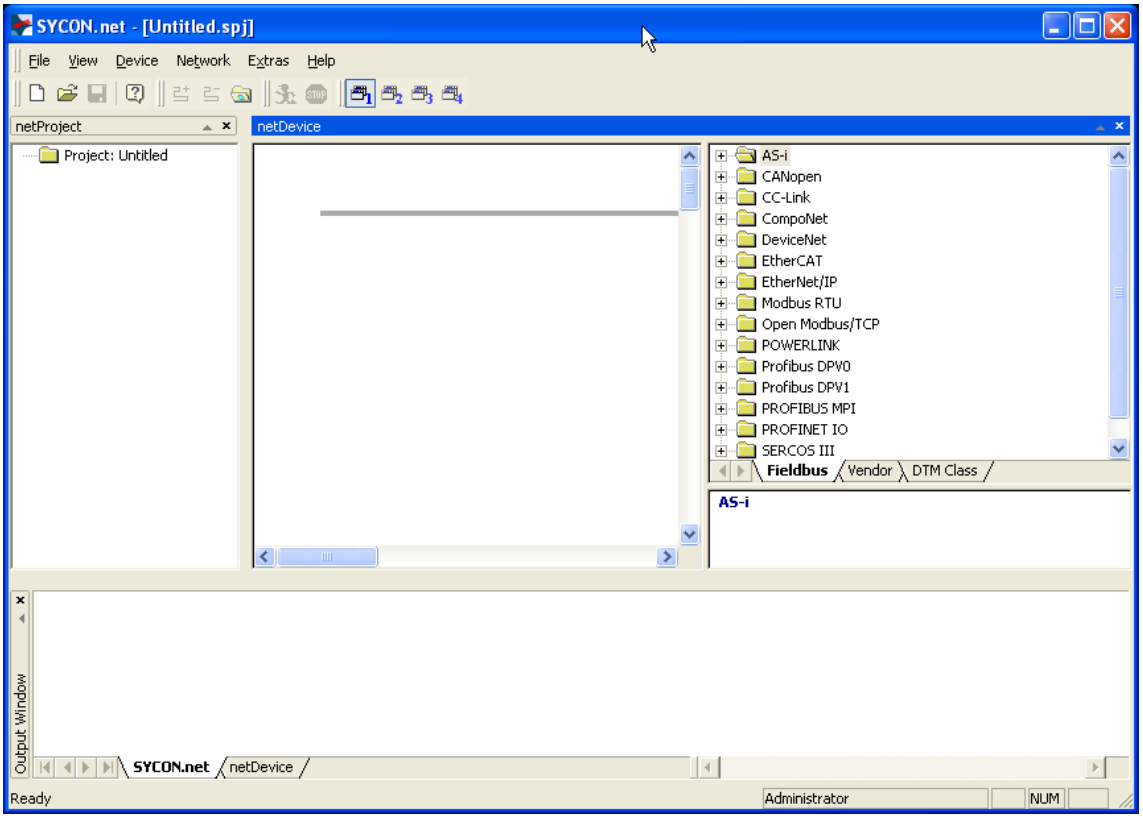
Launch the SYCON.NET software on the PC.



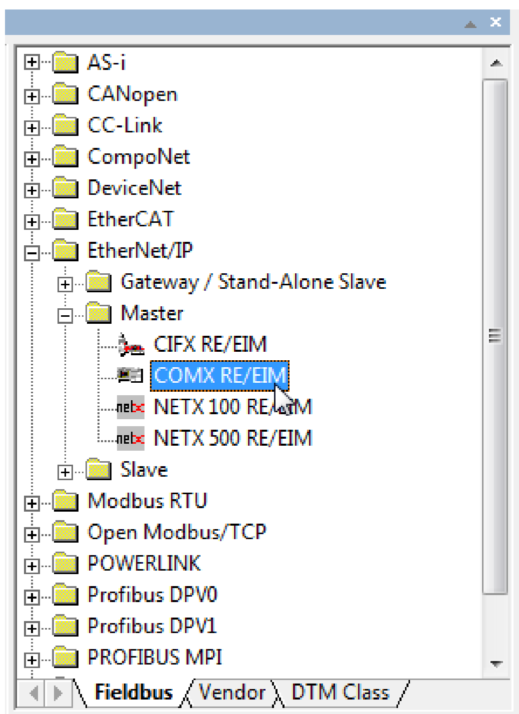
Enter the password:



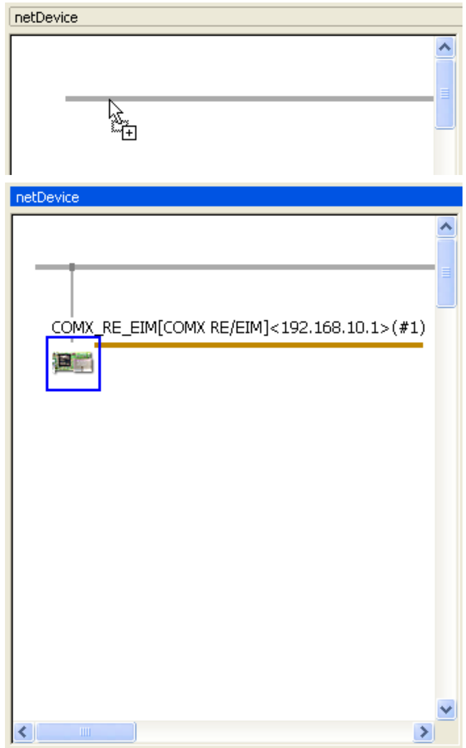
Start a new project or load an existing project from the File menu:



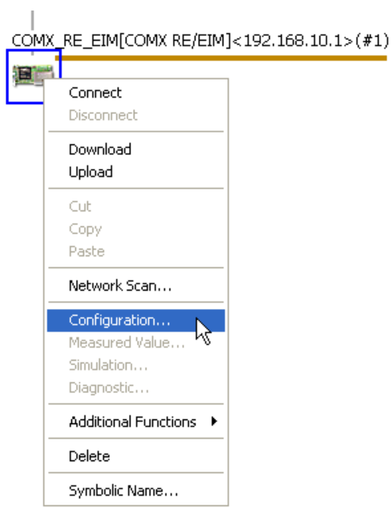
Select the COMX module to which the USB is connected from the Fieldbus protocol list. In this example, an EtherNet/IP module has been selected:



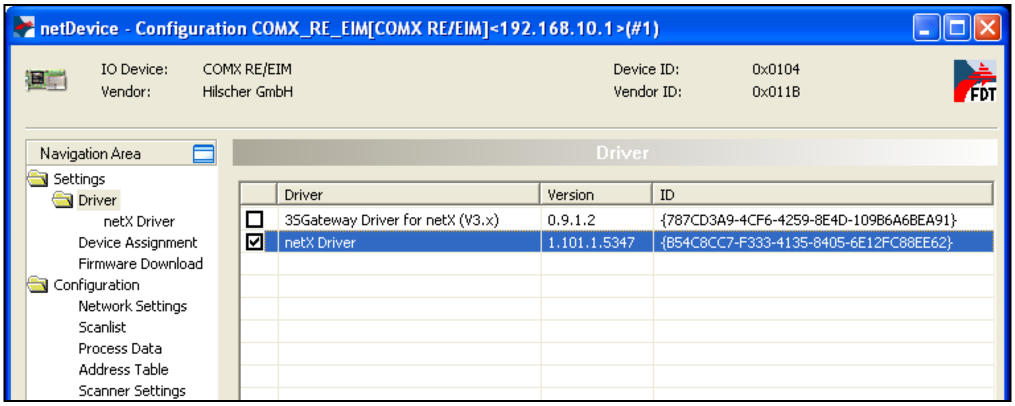
Drag and drop the module onto the BusLine in the netDevice window (notice that the module can only be inserted on the BusLine).



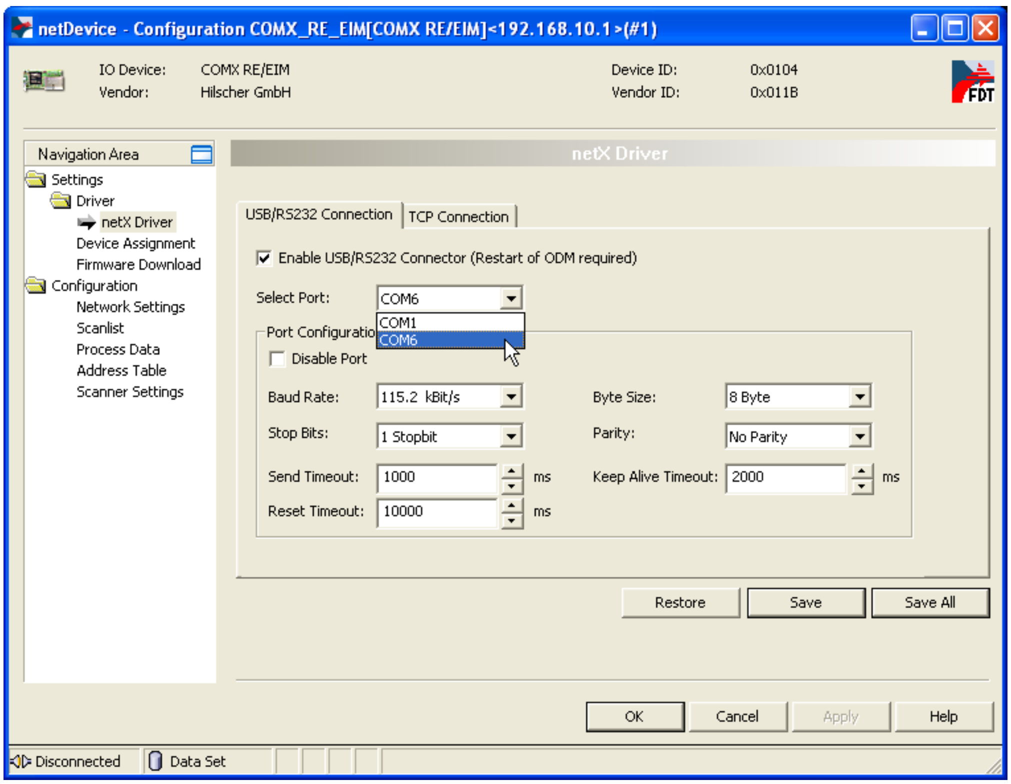
Establish USB communication to the COMX gateway by right-clicking on the device icon and selecting "Configuration…":



In the netDevice Configuration window, select the Driver folder under Settings folder in the NavigationArea, check the checkmark box for netX Driver on the driver list, and click Apply:



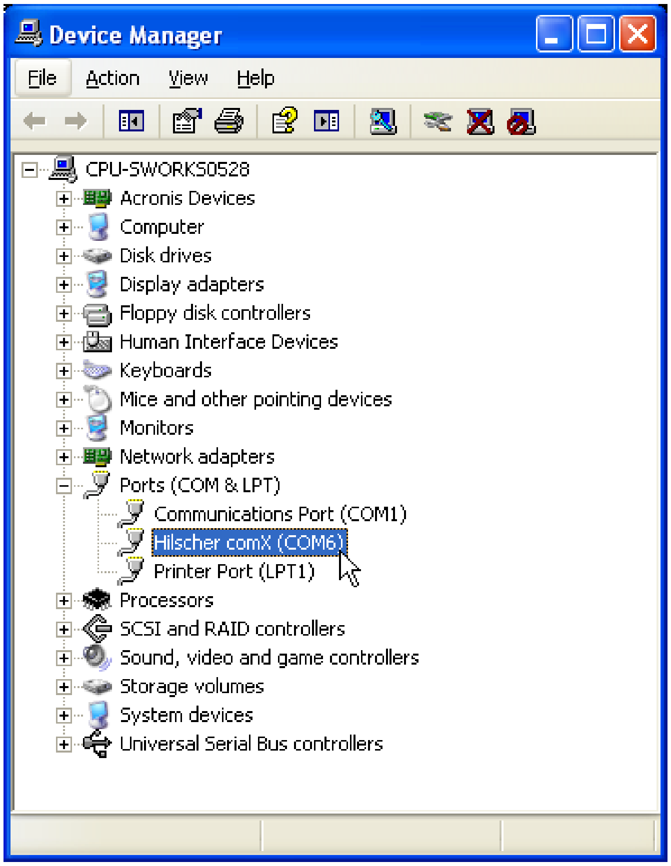
Select the netXDriver node under the Driver folder in the Navigation Area and select the port resembling the USB connection to the COMX module. Click Save and Apply (just click OK if Apply is grayed out).



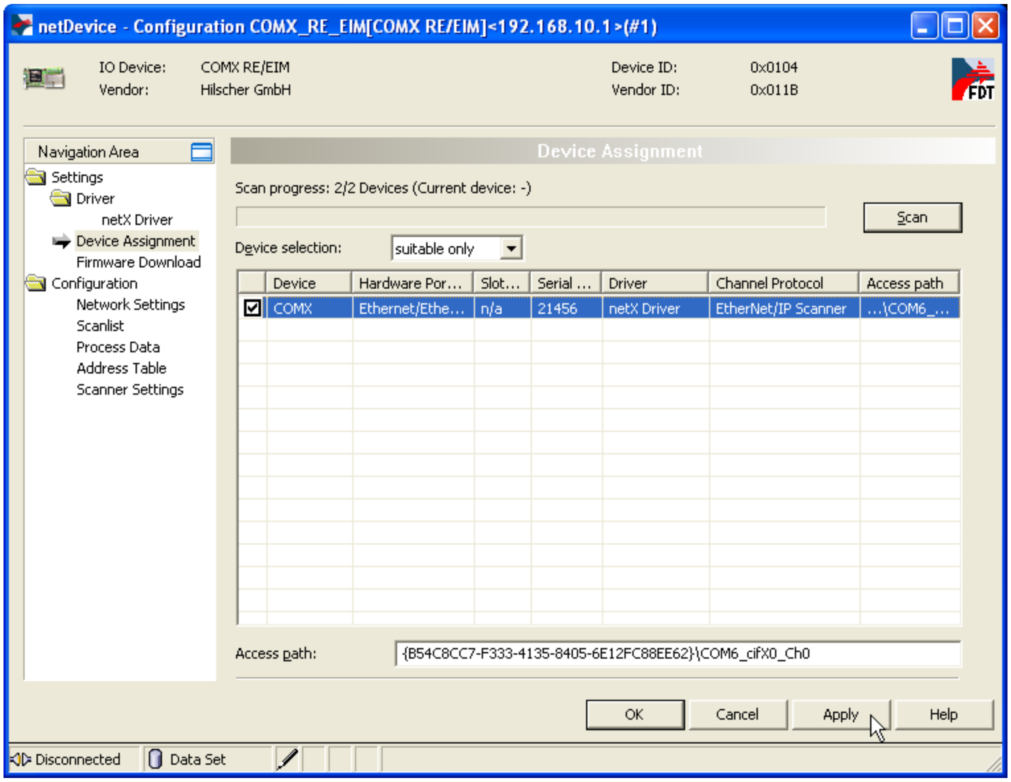


Note

Check Windows Device Manager in order to identify which COM port provides the connection to the Hilscher COMX module.



Click Device Assignment under the Driver folder in the Navigation Area. Assign the netX Driver to the detected COMX module by checking the checkmark box next to the detected device, and click Apply.



Note

When used with Turbo PMAC, the reset line is released too fast for some Hilscher COMX modules, which puts them in a boot mode. This can prevent the device from being detected by Sycon.NET software. Make sure the device receives a system-wide reset using the PMAC suggested M-Variables ulSystemCommandCOS and HSF_RESET registers as shown here.

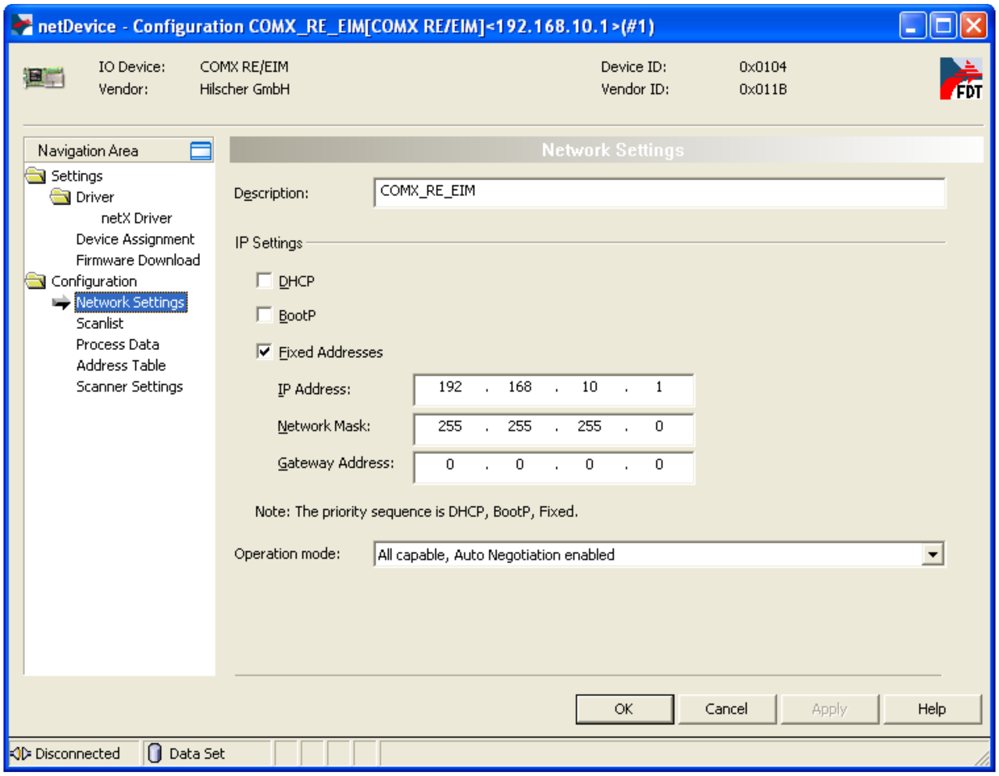
SCtrl_ulSystemCommandCOS=$55AA55AA

HCSC_HSF_RESET=1

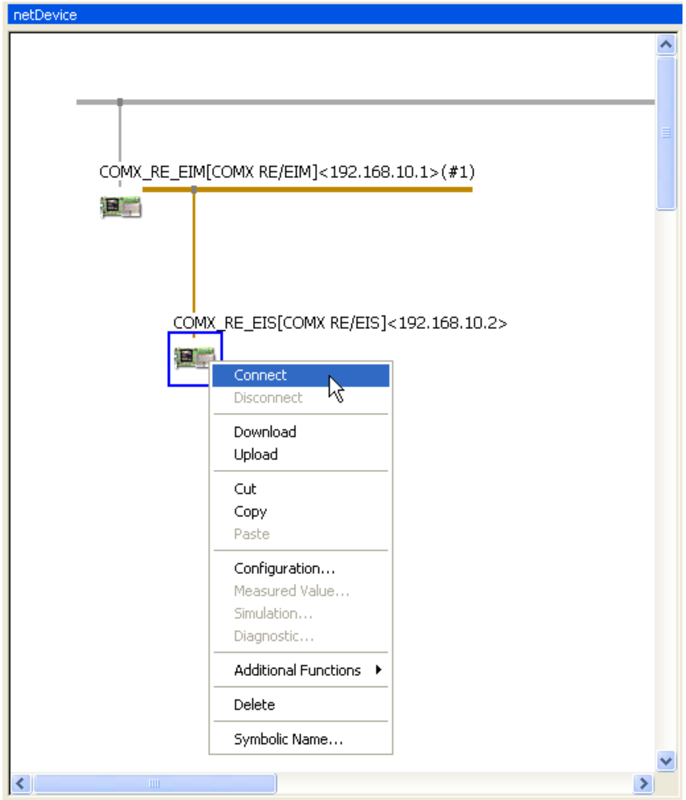
Note that ACC-72EX Setup Assistant software automatically resets the cards if it cannot detect the identification cookie.

The rest of the steps are protocol/module dependent, and it is strongly recommended to follow the directions for these modules in Hilscher documentation available through their website. The current example will be continued with specifics to EtherNet/IP Scanner/Adapter setup.

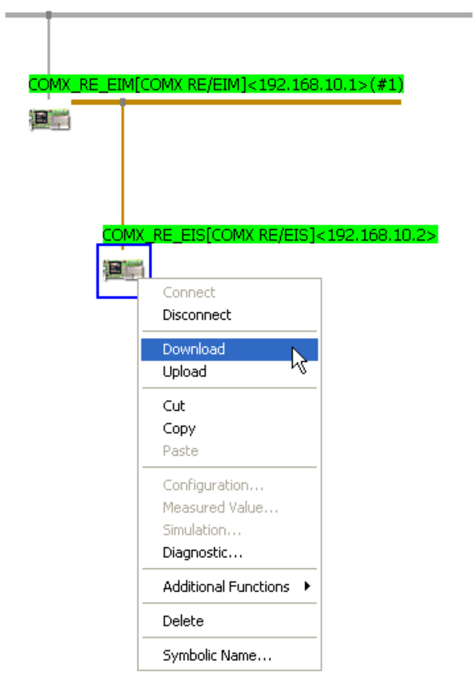
Now that the COMX driver for communication between the PC and COMX module using the diagnostic port has been set up, go through protocol specific setup parameters under the Configuration folder in the Navigation Area.



After finishing modifying the settings for the device, press the OK button.
Back in the netDevice tree, right click on the device icon, and select Connect (as shown below).

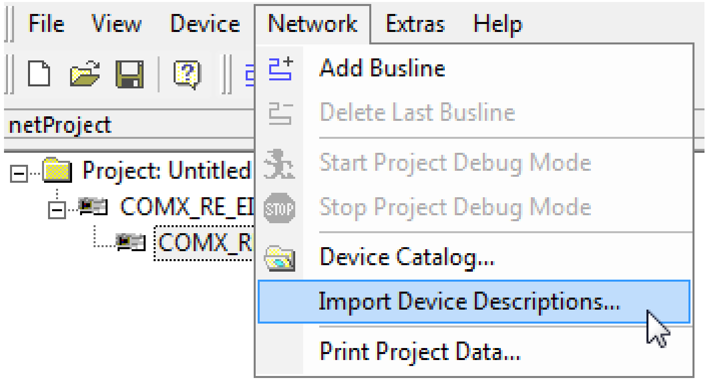


Once connected, right click on the device icon one more time and select Download (as shown below). This will download all the configurations from PC to COMX module.



Once the configuration is downloaded to the COMX module, make sure to save the SYCON.net project for later use.

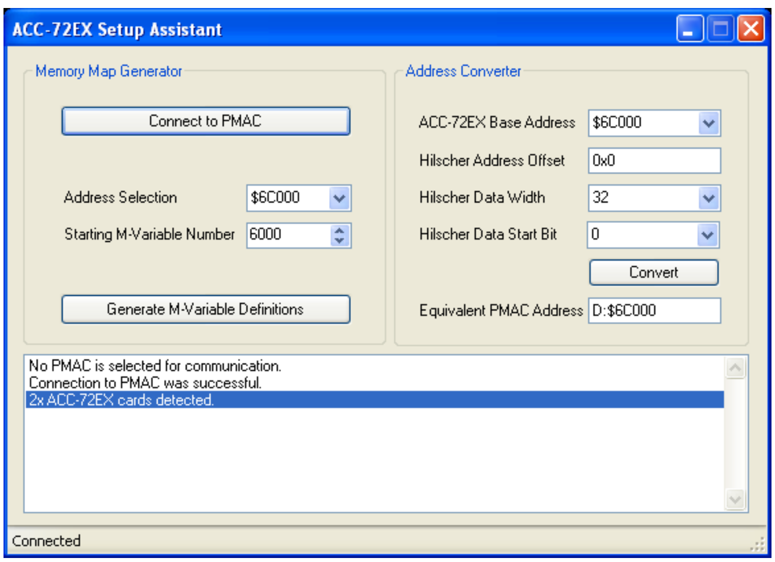
The Hilscher slave module (COMX_RE_IES) above was dragged and dropped from the fieldbus protocol list. Third party slave modules can be added to that list by going to "Import Device Descriptions…" in the Network tab:



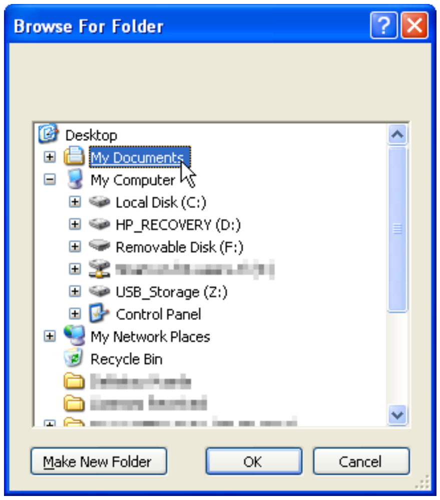
See Appendix C for an example setup using an ACC-72EX Ethernet IP slave with a third party Ethernet IP master PLC controller.

ACC-72EX Setup Assistant

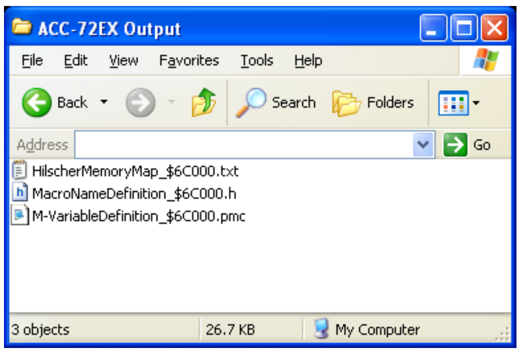
The next step is to generate the memory map and suggested M-Variables for the Hilscher module. Run the ACC-72EX Setup Assistant. In the Memory Map Generator groupbox, click the Connect to PMAC button, and select the UMAC where the ACC-72EX is installed. Once connected, the software will detect any available ACC-72EX(s) in the rack and list it based upon the base address(es).



Select the starting number for M-Variable assignment, and click the"Generate M-Variable Definitions" button. The program asks for a folder location to save the M-Variable definition and memory map files.



This will generate three files which are named based upon the ACC-72EX base address.



The M-variable definition and its header file can be used in writing PLCs and motion programs in PMAC. The memory map file is useful for identifying the process data image locations.

Turbo PMAC Setup for Using ACC-72EX

All interactions between PMAC and ACC-72EX occur through M-variables. Most of the important registers which are required are mapped in the suggested M-Variable definition files generated by ACC-72EX Setup Assistant software. The generated files can be included in the header section of the project files in PEWIN32PRO2.

Note

If multiple ACC-72EX cards are in the same UMAC rack, the M-variable macro names will be identical for both files, despite the file name difference based upon the ACC-72EX base address. Make sure to add a prefix or suffix to the macro names both in the header file and definition file in order to distinguish the proper macro names for different ACC-72EXs. Note that no macro name should be longer than 32 characters.

There are multiple steps in getting the COMX module working with the network/fieldbus. Some of these steps are protocol-specific, and it is recommended to follow the requirements based upon each protocol manual provided by Hilscher.

Initialization PLC

Recall that ACC-72EX requires a reset after each power up, power cycle, $$$ (reset), or $$$*** (factory default reset). This can be achieved with a startup (or initialization) PLC. Example:

```
CLOSE
END GAT
DEL GAT

#include "M-VariableDefinition_$6C000.pmc"
#include "M-VariableDefinition_$74000.pmc"

#define CommErrorFlag P1

OPEN PLC 1 CLEAR
DISABLE PLC 2..31                            // Disable all other tasks
SCtrl_ulSystemCommandCOS=$55AA55AA           // Reset token for MASTER Unit
HCSC_HSF_RESET=1                             // Reset bit, token required for reset to complete
S_SCtrl_ulSystemCommandCOS=$55AA55AA         // Reset token for SLAVE Unit
S_HCSC_HSF_RESET=1                           // Reset bit, token required for reset to complete
CommErrorFlag=0
timer = 1000 msec                            // Reset Time-out Timer
WHILE (CommErrorFlag=0 AND HCSC_NSF_READY=0) // Wait for reset to complete
        IF (timer<0)                         // Check for reset timeout
                CommErrorFlag = 1
        ENDIF
ENDWHILE
IF (CommErrorFlag=0)                                        //
        WHILE (CC0_RCX_COMM_COS_RUN=0 OR S_CC0_RCX_COMM_COS_RUN=0)  // wait for comm tasks to
                                                                    // start on COMX modules
                        HCCC0_HCF_NETX_COS_ACK = HCCC0_HCF_NETX_COS_ACK ^ 1
                        // Toggle Communication Channel 0's Change of State Acknowledge bit in
                        // order to read the CC0_RCX_COMM_COS_RUN which is a part of Communication
                        // Channel 0 State Register
                        S_HCCC0_HCF_NETX_COS_ACK = S_HCCC0_HCF_NETX_COS_ACK ^ 1
        ENDWHILE
        ENABLE PLC 28
        ENABLE PLC 10
        ENABLE PLC 11
ENDIF
DISABLE PLC 1
CLOSE
```

Watchdog Function

The host Watchdog and the device Watchdog cells in the control block of each of the communication channels allow the operating system running on the netX to supervise the host or UMAC application and vice versa. There is no Watchdog function for the system block or for the handshake channel. The Watchdog for the channels is located in the control block of the status block of each communication channel.

The netX firmware reads the contents of the device Watchdog cell, increments the value by one, and copies it back into the host Watchdog location. Then, the application has to copy the new value from the host Watchdog location into the device Watchdog location. Copying the host Watchdog cell to the device Watchdog cell has to happen in the configured Watchdog time. When the overflow occurs, the firmware starts over and “1” appears in the host Watchdog cell. A zero turns off the Watchdog and therefore never appears in the host Watchdog cell in the regular process.

The minimum Watchdog time is 20 ms. The application can start the Watchdog function by copying any value unequal to zero into device Watchdog cell. A zero in the device Watchdog location stops the Watchdog function. The Watchdog timeout is configurable in SYCON.net and can be downloaded to the netX firmware.

If the application fails to copy the value from the host Watchdog location to the device Watchdog location within the configured Watchdog time, the protocol stack will interrupt all network connections immediately, regardless of their current state. If the Watchdog tripped, then power cycling, channel reset, or channel initialization will allow the communication channel to open network connections again.

Here is sample code for copying the host Watchdog location to the device Watchdog location:

```
CLOSE
END GAT
DEL GAT

#include "M-VariableDefinition_$6C000.pmc"
#include "M-VariableDefinition_$74000.pmc"

OPEN PLC 28 CLEAR
CC0_ulDeviceWatchdog = CC0_ulHostWatchdog           // copies the host Watchdog content
                                                    // to device Watchdog cell
                                                    // for the 1st ACC-72EX
S_CC0_ulDeviceWatchdog = S_CC0_ulHostWatchdog       // copies the host Watchdog content
                                                    // to device Watchdog cell
                                                    // for the 2nd ACC-72EX
CLOSE
```

Enabling the Communication Bus

Using the Bus On flag (CCx_RCX_APP_COS_BUS_ON, where x is the communication channel number), the host or UMAC application allows or disallows the netX firmware to open network connections. This flag is used together with the Bus On Enable flag (CCx_RCX_APP_COS_BUS_ON_ENABLE, where x is the communication channel number). If set, the netX firmware tries to open network connections; if cleared, no connections are allowed, and open connections are closed. If the Bus On Enable flag is set, it enables the execution of the Bus On command in the netX firmware:

```
CC0_RCX_APP_COS_BUS_ON=1             // Setting the Bus On flag for 1st ACC-72EX
CC0_RCX_APP_COS_BUS_ON_ENABLE=1      // Enabling the execution of Bus On Flag for 1st ACC-72EX

S_CC0_RCX_APP_COS_BUS_ON=1           // Setting the Bus On flag for 2nd ACC-72EX
S_CC0_RCX_APP_COS_BUS_ON_ENABLE=1    // Enabling the execution of Bus On Flag for 2nd ACC-72EX
```

Locating the Input/Output Data Image in PMAC

Although the ACC-72EX Setup Assistant software defines M-Variables for accessing setup registers and flags in COMX modules, it does not assign any M-Variables for input/output data images. However, starting address and size of each input/output processed data image in's PMAC memory addressing format are calculated and included as a part of the memory map file that is generated. The following is an example from an EtherNet/IP option. The highlighted sections show the addressing for the processed data images:

```
+ Block 2:
 |   Channel Type:                 Communication
 |   Size of Channel:              15616 bytes
 |   Channel Start Address:        $6C0C0
 |   Position of Handshake Cells:  IN HANDSHAKE CHANNEL
 |   Size of Handshake Cells:      16 BITS
 |   NetX Handshake Register:      Y:$6C082,0,16
 |   Host Handshake Register:      X:$6C082,0,16
 |   Communication Class:          SCANNER
 |   Protocol Class:               IO-DEVICE
 |   Conformance Class:            0
 |   Number of Subblocks:             9
 |
 |--- Subblock 0: CONTROL
 |      Size:               8 bytes
 |      Start Offset:       $6C0C2
 |      Transfer Direction: OUT (Host System to netX)
 |      Transfer Type:      DPM (Dual-Port Memory)
 |      Handshake Mode:     UNCONTROLLED
 |      Handshake Bit:      0
 |
 |--- Subblock 1: COMMON STATUS
 |      Size:               64 bytes
 |      Start Offset:       $6C0C4
 |      Transfer Direction: IN (netX to Host System)
 |      Transfer Type:      DPM (Dual-Port Memory)
 |      Handshake Mode:     UNCONTROLLED
 |      Handshake Bit:      0
 |
 |--- Subblock 2: EXTENDED STATUS
 |      Size:               432 bytes
 |      Start Offset:       $6C0D4
 |      Transfer Direction: IN (netX to Host System)
 |      Transfer Type:      DPM (Dual-Port Memory)
 |      Handshake Mode:     UNCONTROLLED
 |      Handshake Bit:      0
 |
 |--- Subblock 3: MAILBOX
 |      Size:               1600 bytes
 |      Start Offset:       $6C140
 |      Transfer Direction: OUT (Host System to netX)
 |      Transfer Type:      DPM (Dual-Port Memory)
 |      Handshake Mode:     BUFFERED, HOST CONTROLLED
 |      Handshake Bit:      4
 |
 |--- Subblock 4: MAILBOX
 |      Size:               1600 bytes
 |      Start Offset:       $6C2D0
 |      Transfer Direction: IN (netX to Host System)
 |      Transfer Type:      DPM (Dual-Port Memory)
 |      Handshake Mode:     UNKNOWN
 |      Handshake Bit:      5
 |
 |--- Subblock 5: PROCESS DATA IMAGE
 |      Size:               5760 bytes
 |      Start Offset:       $6C4C0
 |      Transfer Direction: OUT (Host System to netX)
 |      Transfer Type:      DPM (Dual-Port Memory)
 |      Handshake Mode:     BUFFERED, HOST CONTROLLED
 |      Handshake Bit:      6
```

```
|
|--- Subblock 6: PROCESS DATA IMAGE
|       Size:               5760 bytes
|       Start Offset:       $6CA60
|       Transfer Direction: IN (netX to Host System)
|       Transfer Type:      DPM (Dual-Port Memory)
|       Handshake Mode:     BUFFERED, HOST CONTROLLED
|       Handshake Bit:      7
|
|--- Subblock 7: HIGH PRIORITY DATA IMAGE
|       Size:               64 bytes
|       Start Offset:       $6C460
|       Transfer Direction: OUT (Host System to netX)
|       Transfer Type:      DPM (Dual-Port Memory)
|       Handshake Mode:     BUFFERED, HOST CONTROLLED
|       Handshake Bit:      8
|
|--- Subblock 8: HIGH PRIORITY DATA IMAGE
        Size:               64 bytes
        Start Offset:       $6C470
        Transfer Direction: IN (netX to Host System)
        Transfer Type:      DPM (Dual-Port Memory)
        Handshake Mode:     BUFFERED, HOST CONTROLLED
        Handshake Bit:      9
```

Depending on the protocol, users might be interested in:

- Processed Data Images
- High Priority Data Images
- Mailboxes

Also listed in the memory map are starting address, size of each of these memory blocks, handshake method, and flag.

Reading/Writing from/to Input/Output Data Images

There are two methods for accessing processed data images:

1. Direct M-Variable definition to each register

   This method is useful if the number of I/O data variables is small enough

2. Indirect M-Variable access

   This method is mostly used if the number of I/O data count is greater than a comfortable level which can be handled by the direct M-Variable definition method. Refer to the Turbo PMAC Users Manual for detailed information on how to utilize the indirect addressing method.

This example demonstrates a 16-bit integer register transfer. Notice that only the first 16-bit portion of the integer in P200 will be transferred.

```
CLOSE
END GAT
DEL GAT

#include "M-VariableDefinition_$6C000.pmc"
#include "M-VariableDefinition_$74000.pmc"

#define Master_OutputData1    M2000
#define Master_InputData1     M2001
#define Slave_OutputData1     M2002
#define Slave_InputData1      M2003

Master_OutputData1->Y:$6C4C0,0,16,S          // Pointer to byte 0 and 1 of Output Data Image
                                             // of Communication Channel 0 on Master COMX module
Master_InputData1->Y:$6CA60,0,16,S           // Pointer to byte 0 and 1 of Input Data Image
                                             // of Communication Channel 0 on Master COMX module

Slave_OutputData1->Y:$744C0,0,16,S           // Pointer to byte 0 and 1 of Output Data Image
                                             // of Communication Channel 0 on Slave COMX module
Slave_InputData1->Y:$74A60,0,16,S            // Pointer to byte 0 and 1 of Input Data Image
                                             // of Communication Channel 0 on Slave COMX module

P200=0

OPEN PLC 10 CLEAR
IF (HCCC0_HCF_PD0_OUT_CMD = HCCC0_NCF_PD0_OUT_ACK)   // Making sure the ACK flag matches the CMD
                                                     // flag before writing the value to the
                                                     // output data image register
       P200=P200+1
       Master_OutputData1 = P200                     // Copy the value to register
       HCCC0_HCF_PD0_OUT_CMD = HCCC0_HCF_PD0_OUT_CMD^1      // Toggle the CMD flag (^: XOR)
ENDIF
CLOSE
```

In a similar approach the data can be read from an input data image:

```
OPEN PLC 11 CLEAR
IF (HCCC0_NCF_PD0_IN_CMD = HCCC0_HCF_PD0_IN_ACK)            // If CMD flag and ACK flags are
                                                            // equal, then the input data image
                                                            // register can be read
       P201=M2003                                           // read the input data image register
       HCCC0_HCF_PD0_IN_ACK = HCCC0_HCF_PD0_IN_ACK ^ 1      // toggle the acknowledge bit
                                                            // indicating read completion
EndIF
CLOSE
```

Notice that depending on M-Variable definition, different types of data formats can be transferred over the DPR and network:

```
Mxx->X/Y:$[address],[start],[width],[format]         // Short Word M-Variable Definition
                                                     // access to 1, 4, 8, 16 bits of data
                                                     // is possible
Mxx->DP:$[address]                    // Dual-Ported RAM Fixed-Point M-Variable Definition
Mxx->F:$[address]                     // Dual-Ported RAM Floating-Point M-Variable Definition
```

Power PMAC Setup for Using ACC-72EX

Power PMAC has full support for ACC-72EX and all its fieldbus communication variations. Due to built-in data structures for accessing ACC-72EX dual ported RAM from Power PMAC, no additional software is required for memory mapping and/or identification in comparison to Turbo PMAC.

This section of the manual covers Power PMAC's built in data structures for ACC-72EX in addition to providing examples for header files, start-up and handshaking PLCs.

ACC72EX[*i*]. Non-Saved Data Structures

All of the interactions with ACC-72EX can be achieved through data structures defined specifically for ACC-72EX in Power PMAC firmware. The following structures allow access to the DPRAM in bit, byte, 2-byte and 4-byte wide access modes. The bit-wise read and write is only supported through Acc72EX[*i*].Udata16[*j*] data structure.

Acc72EX[*i*].Data8[*j*]

Description: Dual Ported RAM "unsigned 8-bit integer" data array element

Range: 0 .. 2^8-1

Units: address dependent

Power-on default: address dependent

Acc72Ex[*i*].Data8[*j*] is the "***j***th" unsigned 8-bit integer data array element in the **Acc72EX[*i*]** dual-ported RAM. Each of these elements occupies one byte in the DPRAM, and is located starting at ***j*** addresses past the beginning of the buffer (which is located at the address in **Acc72EX[i].a**). This array is defined based upon the Hilscher ComX memory map.

Index values ***j*** in the square brackets can be integer constants in the range 0 to 524,287, or local L-variables. No expressions or non-integer constants are permitted. The size of the DPRAM is dependent on the ACC-72EX communication option and installed Hilscher ComX module.

Acc72Ex[*i*].Data8[*j*] is located in the same registers as **Acc72Ex[*i*].Idata16[*j/2*]**, **Acc72Ex[*i*].Udata16[*j/2*]**, **Acc72Ex[*i*].Idata32[*j/4*]**, **Acc72Ex[*i*].Idata32[*j/4*]** and **Acc72Ex[*i*].Udata32[*j/4*]**. It is the user's responsibility to prevent possible multiple uses of the same register.

In C, this element should be accessed through the C functions ACC72EX_GetData8 and ACC72EX_SetData8 described later in this manual.

Acc72EX[*i*].Idata16[*j*]

Description: Dual Ported RAM "signed 16-bit integer" data array element

Range: -2^{15} .. 2^{15}-1

Units: address dependent

Power-on default: address dependent

Acc72Ex[*i*].Idata16[*j*] is the "***j***th" signed 16-bit integer data array element in the **Acc72EX[*i*]** dual-ported RAM. Each of these elements occupies two bytes in the DPRAM, and is located starting at ***2*j*** addresses past the beginning of the buffer (which is located at the address in **Acc72EX[*i*].a**). This array is defined based upon the Hilscher ComX memory map.

Index values ***j*** in the square brackets can be integer constants in the range 0 to 262,143, or local L-variables. No expressions or non-integer constants are permitted. The size of the DPRAM is dependent on the ACC-72EX communication option and installed Hilscher ComX module.

Acc72Ex[*i*].Idata16[*j*] is located in the same registers as **Acc72Ex[*i*].Data8[2**j*]** to **Acc72Ex[*i*].Data8[*2*j+1*]**, **Acc72Ex[*i*].Udata16[*j*]**, **Acc72Ex[*i*].Idata32[*j/2*]** and **Acc72Ex[*i*].Udata32[*j/2*]**. It is the user's responsibility to prevent possible multiple uses of the same register.

In C, this element should be accessed through the C functions ACC72EX_GetIdata16 and ACC72EX_SetIdata16 described later in this manual.

Acc72EX[*i*].Udata16[*j*]

Description: Dual Ported RAM "unsigned 16-bit integer" data array element

Range: 0 .. 2^{16}-1

Units: address dependent

Power-on default: address dependent

Acc72Ex[*i*].Udata16[*j*] is the "***j***th" unsigned 16-bit integer data array element in the **Acc72EX[*i*]** dual-ported RAM. Each of these elements occupies two bytes in the DPRAM, and is located starting at ***2*j*** addresses past the beginning of the buffer (which is located at the address in **Acc72EX[i].a**). This array is defined based upon the Hilscher ComX memory map.

Index values ***j*** in the square brackets can be integer constants in the range 0 to 262,143, or local L-variables. No expressions or non-integer constants are permitted. The size of the DPRAM is dependent on the ACC-72EX communication option and installed Hilscher ComX module.

Acc72Ex[*i*].Udata16[*j*] is located in the same registers as **Acc72Ex[*i*].Data8[2**j*]** to **Acc72Ex[*i*].Data8[*2*j+1*], Acc72Ex[*i*].Idata16[*j*]**, **Acc72Ex[*i*].Idata32[*j/2*]** and **Acc72Ex[*i*].Udata32[*j/2*]**. It is the user's responsibility to prevent possible multiple uses of the same register.

In C, this element should be accessed through the C functions ACC72EX_GetUdata16 and ACC72EX_SetUdata16 described later in this manual.

Acc72EX[*i*].Idata32[*j*]

Description: Dual Ported RAM "signed 32-bit integer" data array element

Range: -2^{31} .. 2^{31}-1

Units: address dependent

Power-on default: address dependent

Acc72Ex[*i*].Idata32[*j*] is the "***j***th" signed 32-bit integer data array element in the **Acc72EX[*i*]** dual-ported RAM. Each of these elements occupies four bytes in the DPRAM, and is located starting at ***4*j*** addresses past the beginning of the buffer (which is located at the address in **Acc72EX[*i*].a**). This array is defined based upon the Hilscher ComX memory map.

Index values ***j*** in the square brackets can be integer constants in the range 0 to 131,072, or local L-variables. No expressions or non-integer constants are permitted. The size of the DPRAM is dependent on the ACC-72EX communication option and installed Hilscher ComX module.

Acc72Ex[*i*].Idata32[*j*] is located in the same registers as **Acc72Ex[*i*].Data8[4**j*]** to **Acc72Ex[*i*].Data8[*4*j+5*]**, **Acc72Ex[*i*].Idata16[2**j*]** to **Acc72Ex[*i*].Idata16[2**j+1*]**, **Acc72Ex[*i*].Udata16[2**j*]** to **Acc72Ex[*i*].Udata16[2**j+1*]** and **Acc72Ex[*i*].Udata32[*j*]**. It is the user's responsibility to prevent possible multiple uses of the same register.

In C, this element should be accessed through the C functions ACC72EX_GetIdata32 and ACC72EX_SetIdata32 described later in this manual.

Acc72EX[*i*].Udata32[*j*]

Description: Dual Ported RAM "unsigned 16-bit integer" data array element

Range: 0 .. 2^{32}-1

Units: address dependent

Power-on default: address dependent

Acc72Ex[*i*].Udata32[*j*] is the "***j***th" unsigned 32-bit integer data array element in the **Acc72EX[*i*]** dual ported RAM. Each of these elements occupies four bytes in the DPRAM, and is located starting at ***4*j*** addresses past the beginning of the buffer (which is located at the address in **Acc72EX[*i*].a**). This array is defined based upon the Hilscher ComX memory map.

Index values ***j*** in the square brackets can be integer constants in the range 0 to 262,143, or local L-variables. No expressions or non-integer constants are permitted. The size of the DPRAM is dependent on the ACC-72EX communication option and installed Hilscher ComX module.

Acc72Ex[*i*].Udata32[*j*] is located in the same registers as **Acc72Ex[*i*].Data8[4**j*]** to **Acc72Ex[*i*].Data8[*4*j+5*]**, **Acc72Ex[*i*].Idata16[2**j*]** to **Acc72Ex[*i*].Idata16[2**j+1*]**, **Acc72Ex[*i*].Udata16[2**j*]** to **Acc72Ex[*i*].Udata16[2**j+1*]** and **Acc72Ex[*i*].Idata32[*j*]**. It is the user's responsibility to prevent possible multiple uses of the same register.

In C, this element should be accessed through the C functions ACC72EX_GetUdata32 and ACC72EX_SetUdata32 described later in this manual.

C Programming Access to ACC-72EX Structures

One can use the following header file full of functions to read from and write to the aforementioned **Acc72EX[*i*]** structures from a C program. The input argument CardIndex is *i* and ArrayIndex is *j* as above. Use the "Get" functions to retrieve the structure values; use the "Set" functions to write to the structures. In the "Set" functions, the Input argument is the value to which to set the structure.

```
int Acc72EX_GetIdata32(unsigned int CardIndex, unsigned int ArrayIndex);
unsigned int Acc72EX_GetUdata32(unsigned int CardIndex, unsigned int ArrayIndex);
short Acc72EX_GetIdata16(unsigned int CardIndex, unsigned int ArrayIndex);
char Acc72EX_GetData8(unsigned int CardIndex, unsigned int ArrayIndex);
unsigned short Acc72EX_GetUdata16(unsigned int CardIndex, unsigned int ArrayIndex);

void Acc72EX_SetIdata16(unsigned int CardIndex, unsigned int ArrayIndex, short Input);
void Acc72EX_SetUdata16(unsigned int CardIndex, unsigned int ArrayIndex, unsigned short Input);
void Acc72EX_SetIdata32(unsigned int CardIndex, unsigned int ArrayIndex, int Input);
void Acc72EX_SetUdata32(unsigned int CardIndex, unsigned int ArrayIndex, unsigned int Input);
void Acc72EX_SetData8(unsigned int CardIndex, unsigned int ArrayIndex, char Input);


short Acc72EX_GetIdata16(unsigned int CardIndex, unsigned int ArrayIndex)
{
	unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
	return (short)((myptr[ArrayIndex] << 8) >> 16);
}

unsigned short Acc72EX_GetUdata16(unsigned int CardIndex, unsigned int ArrayIndex)
{
	unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
	return (unsigned short)((myptr[ArrayIndex] << 8) >> 16);
}

char Acc72EX_GetData8(unsigned int CardIndex, unsigned int ArrayIndex)
{
	unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
	return (myptr[ArrayIndex / 2] << (16 / (1 + (ArrayIndex % 4) % 2))) >> 24;
}

unsigned int Acc72EX_GetUdata32(unsigned int CardIndex, unsigned int ArrayIndex)
{
	unsigned int i = ArrayIndex * 4, j, k = 0;
	unsigned int out = 0;

	for(j = i; j <= i + 3; j++)
	{
		out |= (unsigned int)((unsigned int)Acc72EX_GetData8(CardIndex, j) << (8 * k));
		k++;
	}

	return out;
}

int Acc72EX_GetIdata32(unsigned int CardIndex, unsigned int ArrayIndex)
{
	return (int)Acc72EX_GetUdata32(CardIndex, ArrayIndex);
}

void Acc72EX_SetIdata16(unsigned int CardIndex, unsigned int ArrayIndex, short Input)
{
	unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
	myptr[ArrayIndex] = (Input << 8) & 0x00FFFF00;
}

void Acc72EX_SetUdata16(unsigned int CardIndex, unsigned int ArrayIndex, unsigned short Input)
{
	unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
	myptr[ArrayIndex] = (Input << 8) & 0x00FFFF00;
}
```

```
void Acc72EX_SetIdata32(unsigned int CardIndex, unsigned int ArrayIndex, int Input)
{
        unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
        myptr[ArrayIndex] = ((Input << 16) >> 8);
        myptr[ArrayIndex + 1] = ((Input >> 16) << 8);
}

void Acc72EX_SetUdata32(unsigned int CardIndex, unsigned int ArrayIndex, unsigned int Input){
        unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
        myptr[ArrayIndex] = (Input << 16) >> 8;
        myptr[ArrayIndex + 1] = ((Input >> 16) << 8);
}

void Acc72EX_SetData8(unsigned int CardIndex, unsigned int ArrayIndex, char Input){
        unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
        unsigned int shift = (8 * (1 + ArrayIndex % 2));
        unsigned int ind = ArrayIndex / 2;
        myptr[ind] &= ~(0x000000FF << shift);
        myptr[ind] |= (Input << shift);
}
```

Global Header for Power PMAC Projects

This section provides example for header files which allow use of native netX variable names rather than using Power PMAC structures. The following header file is written as generically as possible allowing access to most used registers in System, Handshake and Communication Channels.

```
/*************************************************************************************************/
/* ACC-72EX Power PMAC Project Header
/* This header file provides macro definitions for most common registers in Hilsche COMX modules
/* used in ACC-72EX.
/*
/* Instructions:
/* Uncomment the related #define depending on ACC-72EX option
/*
/*************************************************************************************************/

//#define __PROFIBUS_DP_Master__
//#define __PROFIBUS_DP_Slave__
//#define __DeviceNet_Master__
//#define __DeviceNet_Slave__
//#define __CANopen_Master__
//#define __CANopen_Slave__
//#define __CC_Link_Slave__
//#define __EtherCAT_Master__
//#define __EtherCAT_Slave__
//#define __EtherNetIP_Scanner_Master__
//#define __EtherNetIP_Adapter_Slave__
#define __Open_Modbus_TCP__
//#define __PROFINET_IO_Controller_Master__
//#define __PROFINET_IO_Device_Slave__

// System Information Block Structure
       #define SI_abCookie_0_                           Acc72Ex[0].Data8[0]
       #define SI_abCookie_1_                           Acc72Ex[0].Data8[1]
       #define SI_abCookie_2_                           Acc72Ex[0].Data8[2]
       #define SI_abCookie_3_                           Acc72Ex[0].Data8[3]
       #define SI_ulDpmTotalSize                        Acc72Ex[0].Udata32[1]
       #define SI_ulDeviceNumber                        Acc72Ex[0].Udata32[2]
       #define SI_ulSerialNumber                        Acc72Ex[0].Udata32[3]
       #define SI_ausHwOptions_0_                       Acc72Ex[0].Udata16[8]
       #define SI_ausHwOptions_1_                       Acc72Ex[0].Udata16[9]
       #define SI_ausHwOptions_2_                       Acc72Ex[0].Udata16[10]
       #define SI_ausHwOptions_3_                       Acc72Ex[0].Udata16[11]
       #define SI_usManufacturer                        Acc72Ex[0].Udata16[12]
       #define SI_usProductionDate                      Acc72Ex[0].Udata16[13]
       #define SI_ulLicenseFlags1                       Acc72Ex[0].Udata32[7]
       #define SI_ulLicenseFlags2                       Acc72Ex[0].Udata32[8]
       #define SI_usNetxLicenseID                       Acc72Ex[0].Udata16[18]
       #define SI_usNetxLicenseFlags                    Acc72Ex[0].Udata16[19]
       #define SI_usDeviceClass                         Acc72Ex[0].Udata16[20]
       #define SI_bHwRevision                           Acc72Ex[0].Data8[42]
       #define SI_bHwCompatibility                      Acc72Ex[0].Data8[43]
       #define SI_bDevIdNumber                          Acc72Ex[0].Data8[44]

// System Channel Information Structure
       #define SCI_bChannelType                         Acc72Ex[0].Data8[48]
       #define SCI_bSizePositionOfHandshake             Acc72Ex[0].Data8[50]
       #define SCI_bNumberOfBlocks                      Acc72Ex[0].Data8[51]
       #define SCI_ulSizeOfChannel                      Acc72Ex[0].Udata32[13]
       #define SCI_usSizeOfMailbox                      Acc72Ex[0].Udata16[28]
       #define SCI_usMailboxStartOffset                 Acc72Ex[0].Udata16[29]

// Handshake Channel Information Structure
       #define HCI_bChannelType                         Acc72Ex[0].Data8[64]
       #define HCI_ulSizeOfChannel                      Acc72Ex[0].Udata32[17]

// Communication Channel 0 Information Structure
       #define CC0I_bChannelType                        Acc72Ex[0].Data8[80]
       #define CC0I_bChannelId                          Acc72Ex[0].Data8[81]
       #define CC0I_bSizePositionOfHandshake            Acc72Ex[0].Data8[82]
```

```
      #define CC0I_bNumberOfBlocks                          Acc72Ex[0].Data8[83]
      #define CC0I_ulSizeOfChannel                          Acc72Ex[0].Udata32[21]
      #define CC0I_usCommunicationClass                     Acc72Ex[0].Udata16[44]
      #define CC0I_usProtocolClass                          Acc72Ex[0].Udata16[45]
      #define CC0I_usConformanceClass                       Acc72Ex[0].Udata16[46]

// Communication Channel 1 Information Structure
      #define CC1I_bChannelType                             Acc72Ex[0].Data8[96]
      #define CC1I_bChannelId                               Acc72Ex[0].Data8[97]
      #define CC1I_bSizePositionOfHandshake                 Acc72Ex[0].Data8[98]
      #define CC1I_bNumberOfBlocks                          Acc72Ex[0].Data8[99]
      #define CC1I_ulSizeOfChannel                          Acc72Ex[0].Udata32[25]
      #define CC1I_usCommunicationClass                     Acc72Ex[0].Udata16[52]
      #define CC1I_usProtocolClass                          Acc72Ex[0].Udata16[53]
      #define CC1I_usConformanceClass                       Acc72Ex[0].Udata16[54]

// Communication Channel 0 Information Structure
      #define CC2I_bChannelType                             Acc72Ex[0].Data8[112]
      #define CC2I_bChannelId                               Acc72Ex[0].Data8[113]
      #define CC2I_bSizePositionOfHandshake                 Acc72Ex[0].Data8[114]
      #define CC2I_bNumberOfBlocks                          Acc72Ex[0].Data8[115]
      #define CC2I_ulSizeOfChannel                          Acc72Ex[0].Udata32[29]
      #define CC2I_usCommunicationClass                     Acc72Ex[0].Udata16[60]
      #define CC2I_usProtocolClass                          Acc72Ex[0].Udata16[61]
      #define CC2I_usConformanceClass                       Acc72Ex[0].Udata16[62]

// Communication Channel 1 Information Structure
      #define CC3I_bChannelType                             Acc72Ex[0].Data8[128]
      #define CC3I_bChannelId                               Acc72Ex[0].Data8[129]
      #define CC3I_bSizePositionOfHandshake                 Acc72Ex[0].Data8[130]
      #define CC3I_bNumberOfBlocks                          Acc72Ex[0].Data8[131]
      #define CC3I_ulSizeOfChannel                          Acc72Ex[0].Udata32[33]
      #define CC3I_usCommunicationClass                     Acc72Ex[0].Udata16[68]
      #define CC3I_usProtocolClass                          Acc72Ex[0].Udata16[69]
      #define CC3I_usConformanceClass                       Acc72Ex[0].Udata16[70]

// Application Channel 0 Information Structure
      #define AC0I_bChannelType                             Acc72Ex[0].Data8[144]
      #define AC0I_bChannelId                               Acc72Ex[0].Data8[145]
      #define AC0I_bSizePositionOfHandshake                 Acc72Ex[0].Data8[146]
      #define AC0I_bNumberOfBlocks                          Acc72Ex[0].Data8[147]
      #define AC0I_ulSizeOfChannel                          Acc72Ex[0].Udata32[37]

// Application Channel 1 Information Structure
      #define AC1I_bChannelType                             Acc72Ex[0].Data8[160]
      #define AC1I_bChannelId                               Acc72Ex[0].Data8[161]
      #define AC1I_bSizePositionOfHandshake                 Acc72Ex[0].Data8[162]
      #define AC1I_bNumberOfBlocks                          Acc72Ex[0].Data8[163]
      #define AC1I_ulSizeOfChannel                          Acc72Ex[0].Udata32[41]

// System Control Block Structure
      #define SCtrl_ulSystemCommandCOS                      Acc72Ex[0].Udata32[46]

// System Status Block Structure
      #define SStat_ulSystemCOS                             Acc72Ex[0].Udata32[48]
      #define SStat_ulSystemStatus                          Acc72Ex[0].Udata32[49]
      #define SStat_ulSystemError                           Acc72Ex[0].Udata32[50]
      #define SStat_ulBootError                             Acc72Ex[0].Udata32[51]
      #define SStat_ulTimeSinceStart                        Acc72Ex[0].Udata32[52]
      #define SStat_usCpuLoad                               Acc72Ex[0].Udata16[106]

      #define SStat_ulHWFeatures                            Acc72Ex[0].Udata16[108]
// NETX_SYSTEM_SEND_MAILBOX
      #define SSMB_usPackagesAccepted                       Acc72Ex[0].Udata16[128]

      #define SSMB_ulDest                                   Acc72Ex[0].Udata32[65]
      #define SSMB_ulSrc                                    Acc72Ex[0].Udata32[66]
      #define SSMB_ulDestId                                 Acc72Ex[0].Udata32[67]
      #define SSMB_ulSrcId                                  Acc72Ex[0].Udata32[68]
      #define SSMB_ulLen                                    Acc72Ex[0].Udata32[69]
      #define SSMB_ulId                                     Acc72Ex[0].Udata32[70]
```

```
        #define SSMB_ulState                              Acc72Ex[0].Udata32[71]
        #define SSMB_ulCmd                                Acc72Ex[0].Udata32[72]
        #define SSMB_ulExt                                Acc72Ex[0].Udata32[73]
        #define SSMB_ulRout                               Acc72Ex[0].Udata32[74]

ptr SSMB_Data8(84)->*;
ptr SSMB_Data16(42)->*;
ptr SSMB_Data32(21)->*;

// NETX_SYSTEM_RECEIVE_MAILBOX
        #define SRMB_usWaitingPackages                    Acc72Ex[0].Udata16[192]

        #define SRMB_ulDest                               Acc72Ex[0].Udata32[97]
        #define SRMB_ulSrc                                Acc72Ex[0].Udata32[98]
        #define SRMB_ulDestId                             Acc72Ex[0].Udata32[99]
        #define SRMB_ulSrcId                              Acc72Ex[0].Udata32[100]
        #define SRMB_ulLen                                Acc72Ex[0].Udata32[101]
        #define SRMB_ulId                                 Acc72Ex[0].Udata32[102]
        #define SRMB_ulState                              Acc72Ex[0].Udata32[103]
        #define SRMB_ulCmd                                Acc72Ex[0].Udata32[104]
        #define SRMB_ulExt                                Acc72Ex[0].Udata32[105]
        #define SRMB_ulRout                               Acc72Ex[0].Udata32[106]

ptr SRMB_Data8(84)->*;
ptr SRMB_Data16(42)->*;
ptr SRMB_Data32(21)->*;

// SC_bNetxFlags
        #define HCSC_NSF_READY                            Acc72Ex[0].Udata16[257].0
        #define HCSC_NSF_ERROR                            Acc72Ex[0].Udata16[257].1
        #define HCSC_NSF_HOST_COS_ACK                     Acc72Ex[0].Udata16[257].2
        #define HCSC_NSF_NETX_COS_CMD                     Acc72Ex[0].Udata16[257].3
        #define HCSC_NSF_SEND_MBX_ACK                     Acc72Ex[0].Udata16[257].4
        #define HCSC_NSF_RECV_MBX_CMD                     Acc72Ex[0].Udata16[257].5
// SC_bHostFlags
        #define HCSC_HSF_RESET                            Acc72Ex[0].Udata16[257].8
        #define HCSC_HSF_BOOTSTART                        Acc72Ex[0].Udata16[257].9
        #define HCSC_HSF_HOST_COS_CMD                     Acc72Ex[0].Udata16[257].10
        #define HCSC_HSF_NETX_COS_ACK                     Acc72Ex[0].Udata16[257].11
        #define HCSC_HSF_SEND_MBX_CMD                     Acc72Ex[0].Udata16[257].12
        #define HCSC_HSF_RECV_MBX_ACK                     Acc72Ex[0].Udata16[257].13
// CC0 usNetxFlags
        #define HCCC0_usNetxFlags                         Acc72Ex[0].Udata16[260]
        #define HCCC0_NCF_COMMUNICATING                   Acc72Ex[0].Udata16[260].0
        #define HCCC0_NCF_ERROR                           Acc72Ex[0].Udata16[260].1
        #define HCCC0_NCF_HOST_COS_ACK                    Acc72Ex[0].Udata16[260].2
        #define HCCC0_NCF_NETX_COS_CMD                    Acc72Ex[0].Udata16[260].3
        #define HCCC0_NCF_SEND_MBX_ACK                    Acc72Ex[0].Udata16[260].4
        #define HCCC0_NCF_RECV_MBX_CMD                    Acc72Ex[0].Udata16[260].5
        #define HCCC0_NCF_PD0_OUT_ACK                     Acc72Ex[0].Udata16[260].6
        #define HCCC0_NCF_PD0_IN_CMD                      Acc72Ex[0].Udata16[260].7
        #define HCCC0_NCF_PD1_OUT_ACK                     Acc72Ex[0].Udata16[260].8
        #define HCCC0_NCF_PD1_IN_CMD                      Acc72Ex[0].Udata16[260].9
// CC0 usHostFlags
        #define HCCC0_usHostFlags                         Acc72Ex[0].Udata16[261]
        #define HCCC0_HCF_HOST_COS_CMD                    Acc72Ex[0].Udata16[261].2
        #define HCCC0_HCF_NETX_COS_ACK                    Acc72Ex[0].Udata16[261].3
        #define HCCC0_HCF_SEND_MBX_CMD                    Acc72Ex[0].Udata16[261].4
        #define HCCC0_HCF_RECV_MBX_ACK                    Acc72Ex[0].Udata16[261].5
        #define HCCC0_HCF_PD0_OUT_CMD                     Acc72Ex[0].Udata16[261].6
        #define HCCC0_HCF_PD0_IN_ACK                      Acc72Ex[0].Udata16[261].7
        #define HCCC0_HCF_PD1_OUT_CMD                     Acc72Ex[0].Udata16[261].8
        #define HCCC0_HCF_PD1_IN_ACK                      Acc72Ex[0].Udata16[261].9
// CC1 usNetxFlags
        #define HCCC1_usNetxFlags                         Acc72Ex[0].Udata16[262]
        #define HCCC1_NCF_COMMUNICATING                   Acc72Ex[0].Udata16[262].0
        #define HCCC1_NCF_ERROR                           Acc72Ex[0].Udata16[262].1
        #define HCCC1_NCF_HOST_COS_ACK                    Acc72Ex[0].Udata16[262].2
        #define HCCC1_NCF_NETX_COS_CMD                    Acc72Ex[0].Udata16[262].3
        #define HCCC1_NCF_SEND_MBX_ACK                    Acc72Ex[0].Udata16[262].4
        #define HCCC1_NCF_RECV_MBX_CMD                    Acc72Ex[0].Udata16[262].5
```

```
        #define HCCC1_NCF_PD0_OUT_ACK                     Acc72Ex[0].Udata16[262].6
        #define HCCC1_NCF_PD0_IN_CMD                      Acc72Ex[0].Udata16[262].7
        #define HCCC1_NCF_PD1_OUT_ACK                     Acc72Ex[0].Udata16[262].8
        #define HCCC1_NCF_PD1_IN_CMD                      Acc72Ex[0].Udata16[262].9
// CC1 usHostFlags
        #define HCCC1_usHostFlags                         Acc72Ex[0].Udata16[263]
        #define HCCC1_HCF_HOST_COS_CMD                    Acc72Ex[0].Udata16[263].2
        #define HCCC1_HCF_NETX_COS_ACK                    Acc72Ex[0].Udata16[263].3
        #define HCCC1_HCF_SEND_MBX_CMD                    Acc72Ex[0].Udata16[263].4
        #define HCCC1_HCF_RECV_MBX_ACK                    Acc72Ex[0].Udata16[263].5
        #define HCCC1_HCF_PD0_OUT_CMD                     Acc72Ex[0].Udata16[263].6
        #define HCCC1_HCF_PD0_IN_ACK                      Acc72Ex[0].Udata16[263].7
        #define HCCC1_HCF_PD1_OUT_CMD                     Acc72Ex[0].Udata16[263].8
        #define HCCC1_HCF_PD1_IN_ACK                      Acc72Ex[0].Udata16[263].9
// CC2 usNetxFlags
        #define HCCC2_usNetxFlags                         Acc72Ex[0].Udata16[264]
        #define HCCC2_NCF_COMMUNICATING                   Acc72Ex[0].Udata16[264].0
        #define HCCC2_NCF_ERROR                           Acc72Ex[0].Udata16[264].1
        #define HCCC2_NCF_HOST_COS_ACK                    Acc72Ex[0].Udata16[264].2
        #define HCCC2_NCF_NETX_COS_CMD                    Acc72Ex[0].Udata16[264].3
        #define HCCC2_NCF_SEND_MBX_ACK                    Acc72Ex[0].Udata16[264].4
        #define HCCC2_NCF_RECV_MBX_CMD                    Acc72Ex[0].Udata16[264].5
        #define HCCC2_NCF_PD0_OUT_ACK                     Acc72Ex[0].Udata16[264].6
        #define HCCC2_NCF_PD0_IN_CMD                      Acc72Ex[0].Udata16[264].7
        #define HCCC2_NCF_PD1_OUT_ACK                     Acc72Ex[0].Udata16[264].8
        #define HCCC2_NCF_PD1_IN_CMD                      Acc72Ex[0].Udata16[264].9
// CC2 usHostFlags
        #define HCCC2_usHostFlags                         Acc72Ex[0].Udata16[265]
        #define HCCC2_HCF_HOST_COS_CMD                    Acc72Ex[0].Udata16[265].2
        #define HCCC2_HCF_NETX_COS_ACK                    Acc72Ex[0].Udata16[265].3
        #define HCCC2_HCF_SEND_MBX_CMD                    Acc72Ex[0].Udata16[265].4
        #define HCCC2_HCF_RECV_MBX_ACK                    Acc72Ex[0].Udata16[265].5
        #define HCCC2_HCF_PD0_OUT_CMD                     Acc72Ex[0].Udata16[265].6
        #define HCCC2_HCF_PD0_IN_ACK                      Acc72Ex[0].Udata16[265].7
        #define HCCC2_HCF_PD1_OUT_CMD                     Acc72Ex[0].Udata16[265].8
        #define HCCC2_HCF_PD1_IN_ACK                      Acc72Ex[0].Udata16[265].9
// CC3 usNetxFlags
        #define HCCC3_usNetxFlags                         Acc72Ex[0].Udata16[266]
        #define HCCC3_NCF_COMMUNICATING                   Acc72Ex[0].Udata16[266].0
        #define HCCC3_NCF_ERROR                           Acc72Ex[0].Udata16[266].1
        #define HCCC3_NCF_HOST_COS_ACK                    Acc72Ex[0].Udata16[266].2
        #define HCCC3_NCF_NETX_COS_CMD                    Acc72Ex[0].Udata16[266].3
        #define HCCC3_NCF_SEND_MBX_ACK                    Acc72Ex[0].Udata16[266].4
        #define HCCC3_NCF_RECV_MBX_CMD                    Acc72Ex[0].Udata16[266].5
        #define HCCC3_NCF_PD0_OUT_ACK                     Acc72Ex[0].Udata16[266].6
        #define HCCC3_NCF_PD0_IN_CMD                      Acc72Ex[0].Udata16[266].7
        #define HCCC3_NCF_PD1_OUT_ACK                     Acc72Ex[0].Udata16[266].8
        #define HCCC3_NCF_PD1_IN_CMD                      Acc72Ex[0].Udata16[266].9
// CC3 usHostFlags
        #define HCCC3_usHostFlags                         Acc72Ex[0].Udata16[267]
        #define HCCC3_HCF_HOST_COS_CMD                    Acc72Ex[0].Udata16[267].2
        #define HCCC3_HCF_NETX_COS_ACK                    Acc72Ex[0].Udata16[267].3
        #define HCCC3_HCF_SEND_MBX_CMD                    Acc72Ex[0].Udata16[267].4
        #define HCCC3_HCF_RECV_MBX_ACK                    Acc72Ex[0].Udata16[267].5
        #define HCCC3_HCF_PD0_OUT_CMD                     Acc72Ex[0].Udata16[267].6
        #define HCCC3_HCF_PD0_IN_ACK                      Acc72Ex[0].Udata16[267].7
        #define HCCC3_HCF_PD1_OUT_CMD                     Acc72Ex[0].Udata16[267].8
        #define HCCC3_HCF_PD1_IN_ACK                      Acc72Ex[0].Udata16[267].9

// CC0_Control Block
        #define CC0_RCX_APP_COS_APP_READY                 Acc72Ex[0].Udata16[388].0
        #define CC0_RCX_APP_COS_BUS_ON                    Acc72Ex[0].Udata16[388].1
        #define CC0_RCX_APP_COS_BUS_ON_ENABLE             Acc72Ex[0].Udata16[388].2
        #define CC0_RCX_APP_COS_INIT                      Acc72Ex[0].Udata16[388].3
        #define CC0_RCX_APP_COS_INIT_ENABLE               Acc72Ex[0].Udata16[388].4
        #define CC0_RCX_APP_COS_LOCK_CFG                  Acc72Ex[0].Udata16[388].5
        #define CC0_RCX_APP_COS_LOCK_CFG_ENA              Acc72Ex[0].Udata16[388].6
        #define CC0_RCX_APP_COS_DMA                       Acc72Ex[0].Udata16[388].7
        #define CC0_RCX_APP_COS_DMA_ENABLE                Acc72Ex[0].Udata16[388].8

        #define CC0_ulDeviceWatchdog                      Acc72Ex[0].Udata32[195]
```

```
// CC0_CommunicationCOS
       #define CC0_RCX_COMM_COS_READY                      Acc72Ex[0].Udata16[392].0
       #define CC0_RCX_COMM_COS_RUN                        Acc72Ex[0].Udata16[392].1
       #define CC0_RCX_COMM_COS_BUS_ON                     Acc72Ex[0].Udata16[392].2
       #define CC0_RCX_COMM_COS_CONFIG_LOCKED              Acc72Ex[0].Udata16[392].3
       #define CC0_RCX_COMM_COS_CONFIG_NEW                 Acc72Ex[0].Udata16[392].4
       #define CC0_RCX_COMM_COS_RESTART_REQ                Acc72Ex[0].Udata16[392].5
       #define CC0_RCX_COMM_COS_RESTART_REQ_ENA            Acc72Ex[0].Udata16[392].6
       #define CC0_RCX_COMM_COS_DMA                        Acc72Ex[0].Udata16[392].7

// CC0_Status Block
       #define CC0_ulCommunicationState                    Acc72Ex[0].Udata32[197]
       #define CC0_ulCommunicationError                    Acc72Ex[0].Udata32[198]
       #define CC0_usVersion                               Acc72Ex[0].Udata16[398]
       #define CC0_usWatchdogTime                          Acc72Ex[0].Udata16[399]
       #define CC0_bPDInHskMode                            Acc72Ex[0].Data8[800]
       #define CC0_bPDInSource                             Acc72Ex[0].Data8[801]
       #define CC0_bPDOutHskMode                           Acc72Ex[0].Data8[802]
       #define CC0_bPDOutSource                            Acc72Ex[0].Data8[803]
       #define CC0_ulHostWatchdog                          Acc72Ex[0].Udata32[201]
       #define CC0_ulErrorCount                            Acc72Ex[0].Udata32[202]
       #define CC0_bErrorLogInd                            Acc72Ex[0].Data8[812]
       #define CC0_bErrorPDInCnt                           Acc72Ex[0].Data8[813]
       #define CC0_bErrorPDOutCnt                          Acc72Ex[0].Data8[814]
       #define CC0_bErrorSyncCnt                           Acc72Ex[0].Data8[815]
       #define CC0_bSyncHskMode                            Acc72Ex[0].Data8[816]
       #define CC0_bSyncSource                             Acc72Ex[0].Data8[817]

// CC1_Control Block
       #define CC1_RCX_APP_COS_APP_READY                   Acc72Ex[0].Udata16[8196].0
       #define CC1_RCX_APP_COS_BUS_ON                      Acc72Ex[0].Udata16[8196].1
       #define CC1_RCX_APP_COS_BUS_ON_ENABLE               Acc72Ex[0].Udata16[8196].2
       #define CC1_RCX_APP_COS_INIT                        Acc72Ex[0].Udata16[8196].3
       #define CC1_RCX_APP_COS_INIT_ENABLE                 Acc72Ex[0].Udata16[8196].4
       #define CC1_RCX_APP_COS_LOCK_CFG                    Acc72Ex[0].Udata16[8196].5
       #define CC1_RCX_APP_COS_LOCK_CFG_ENA                Acc72Ex[0].Udata16[8196].6
       #define CC1_RCX_APP_COS_DMA                         Acc72Ex[0].Udata16[8196].7
       #define CC1_RCX_APP_COS_DMA_ENABLE                  Acc72Ex[0].Udata16[8196].8

       #define CC1_ulDeviceWatchdog                        Acc72Ex[0].Udata32[4099]
// CC1_CommunicationCOS
       #define CC1_RCX_COMM_COS_READY                      Acc72Ex[0].Udata16[8200].0
       #define CC1_RCX_COMM_COS_RUN                        Acc72Ex[0].Udata16[8200].1
       #define CC1_RCX_COMM_COS_BUS_ON                     Acc72Ex[0].Udata16[8200].2
       #define CC1_RCX_COMM_COS_CONFIG_LOCKED              Acc72Ex[0].Udata16[8200].3
       #define CC1_RCX_COMM_COS_CONFIG_NEW                 Acc72Ex[0].Udata16[8200].4
       #define CC1_RCX_COMM_COS_RESTART_REQ                Acc72Ex[0].Udata16[8200].5
       #define CC1_RCX_COMM_COS_RESTART_REQ_ENA            Acc72Ex[0].Udata16[8200].6
       #define CC1_RCX_COMM_COS_DMA                        Acc72Ex[0].Udata16[8200].7

// CC1_Status Block
       #define CC1_ulCommunicationState                    Acc72Ex[0].Udata32[4101]
       #define CC1_ulCommunicationError                    Acc72Ex[0].Udata32[4102]
       #define CC1_usVersion                               Acc72Ex[0].Udata16[8206]
       #define CC1_usWatchdogTime                          Acc72Ex[0].Udata16[8207]
       #define CC1_bPDInHskMode                            Acc72Ex[0].Data8[16416]
       #define CC1_bPDInSource                             Acc72Ex[0].Data8[16417]
       #define CC1_bPDOutHskMode                           Acc72Ex[0].Data8[16418]
       #define CC1_bPDOutSource                            Acc72Ex[0].Data8[16419]
       #define CC1_ulHostWatchdog                          Acc72Ex[0].Udata32[4105]
       #define CC1_ulErrorCount                            Acc72Ex[0].Udata32[4106]
       #define CC1_bErrorLogInd                            Acc72Ex[0].Data8[16428]
       #define CC1_bErrorPDInCnt                           Acc72Ex[0].Data8[16429]
       #define CC1_bErrorPDOutCnt                          Acc72Ex[0].Data8[16430]
       #define CC1_bErrorSyncCnt                           Acc72Ex[0].Data8[16431]
       #define CC1_bSyncHskMode                            Acc72Ex[0].Data8[16432]
       #define CC1_bSyncSource                             Acc72Ex[0].Data8[16433]

// CC2_Control Block
       #define CC2_RCX_APP_COS_APP_READY                   Acc72Ex[0].Udata16[16004].0
       #define CC2_RCX_APP_COS_BUS_ON                      Acc72Ex[0].Udata16[16004].1
```

```
	#define CC2_RCX_APP_COS_BUS_ON_ENABLE		Acc72Ex[0].Udata16[16004].2
	#define CC2_RCX_APP_COS_INIT			Acc72Ex[0].Udata16[16004].3
	#define CC2_RCX_APP_COS_INIT_ENABLE		Acc72Ex[0].Udata16[16004].4
	#define CC2_RCX_APP_COS_LOCK_CFG		Acc72Ex[0].Udata16[16004].5
	#define CC2_RCX_APP_COS_LOCK_CFG_ENA		Acc72Ex[0].Udata16[16004].6
	#define CC2_RCX_APP_COS_DMA			Acc72Ex[0].Udata16[16004].7
	#define CC2_RCX_APP_COS_DMA_ENABLE		Acc72Ex[0].Udata16[16004].8

	#define CC2_ulDeviceWatchdog			Acc72Ex[0].Udata32[8003]
// CC2_CommunicationCOS
	#define CC2_RCX_COMM_COS_READY			Acc72Ex[0].Udata16[16008].0
	#define CC2_RCX_COMM_COS_RUN			Acc72Ex[0].Udata16[16008].1
	#define CC2_RCX_COMM_COS_BUS_ON			Acc72Ex[0].Udata16[16008].2
	#define CC2_RCX_COMM_COS_CONFIG_LOCKED		Acc72Ex[0].Udata16[16008].3
	#define CC2_RCX_COMM_COS_CONFIG_NEW		Acc72Ex[0].Udata16[16008].4
	#define CC2_RCX_COMM_COS_RESTART_REQ		Acc72Ex[0].Udata16[16008].5
	#define CC2_RCX_COMM_COS_RESTART_REQ_ENA	Acc72Ex[0].Udata16[16008].6
	#define CC2_RCX_COMM_COS_DMA			Acc72Ex[0].Udata16[16008].7

// CC2_Status Block
	#define CC2_ulCommunicationState		Acc72Ex[0].Udata32[8005]
	#define CC2_ulCommunicationError		Acc72Ex[0].Udata32[8006]
	#define CC2_usVersion				Acc72Ex[0].Udata16[16014]
	#define CC2_usWatchdogTime			Acc72Ex[0].Udata16[16015]
	#define CC2_bPDInHskMode			Acc72Ex[0].Data8[32032]
	#define CC2_bPDInSource				Acc72Ex[0].Data8[32033]
	#define CC2_bPDOutHskMode			Acc72Ex[0].Data8[32034]
	#define CC2_bPDOutSource			Acc72Ex[0].Data8[32035]
	#define CC2_ulHostWatchdog			Acc72Ex[0].Udata32[8009]
	#define CC2_ulErrorCount			Acc72Ex[0].Udata32[8010]
	#define CC2_bErrorLogInd			Acc72Ex[0].Data8[32044]
	#define CC2_bErrorPDInCnt			Acc72Ex[0].Data8[32045]
	#define CC2_bErrorPDOutCnt			Acc72Ex[0].Data8[32046]
	#define CC2_bErrorSyncCnt			Acc72Ex[0].Data8[32047]
	#define CC2_bSyncHskMode			Acc72Ex[0].Data8[32048]
	#define CC2_bSyncSource				Acc72Ex[0].Data8[32049]

// CC3_Control Block
	#define CC3_RCX_APP_COS_APP_READY		Acc72Ex[0].Udata16[23812].0
	#define CC3_RCX_APP_COS_BUS_ON			Acc72Ex[0].Udata16[23812].1
	#define CC3_RCX_APP_COS_BUS_ON_ENABLE		Acc72Ex[0].Udata16[23812].2
	#define CC3_RCX_APP_COS_INIT			Acc72Ex[0].Udata16[23812].3
	#define CC3_RCX_APP_COS_INIT_ENABLE		Acc72Ex[0].Udata16[23812].4
	#define CC3_RCX_APP_COS_LOCK_CFG		Acc72Ex[0].Udata16[23812].5
	#define CC3_RCX_APP_COS_LOCK_CFG_ENA		Acc72Ex[0].Udata16[23812].6
	#define CC3_RCX_APP_COS_DMA			Acc72Ex[0].Udata16[23812].7
	#define CC3_RCX_APP_COS_DMA_ENABLE		Acc72Ex[0].Udata16[23812].8

	#define CC3_ulDeviceWatchdog			Acc72Ex[0].Udata32[11907]
// CC3_CommunicationCOS
	#define CC3_RCX_COMM_COS_READY			Acc72Ex[0].Udata16[23816].0
	#define CC3_RCX_COMM_COS_RUN			Acc72Ex[0].Udata16[23816].1
	#define CC3_RCX_COMM_COS_BUS_ON			Acc72Ex[0].Udata16[23816].2
	#define CC3_RCX_COMM_COS_CONFIG_LOCKED		Acc72Ex[0].Udata16[23816].3
	#define CC3_RCX_COMM_COS_CONFIG_NEW		Acc72Ex[0].Udata16[23816].4
	#define CC3_RCX_COMM_COS_RESTART_REQ		Acc72Ex[0].Udata16[23816].5
	#define CC3_RCX_COMM_COS_RESTART_REQ_ENA	Acc72Ex[0].Udata16[23816].6
	#define CC3_RCX_COMM_COS_DMA			Acc72Ex[0].Udata16[23816].7

// CC3_Status Block
	#define CC3_ulCommunicationState		Acc72Ex[0].Udata32[11909]
	#define CC3_ulCommunicationError		Acc72Ex[0].Udata32[11910]
	#define CC3_usVersion				Acc72Ex[0].Udata16[23822]
	#define CC3_usWatchdogTime			Acc72Ex[0].Udata16[23823]
	#define CC3_bPDInHskMode			Acc72Ex[0].Data8[47648]
	#define CC3_bPDInSource				Acc72Ex[0].Data8[47649]
	#define CC3_bPDOutHskMode			Acc72Ex[0].Data8[47650]
	#define CC3_bPDOutSource			Acc72Ex[0].Data8[47651]
	#define CC3_ulHostWatchdog			Acc72Ex[0].Udata32[11913]
	#define CC3_ulErrorCount			Acc72Ex[0].Udata32[11914]
	#define CC3_bErrorLogInd			Acc72Ex[0].Data8[47660]
```

```
        #define CC3_bErrorPDInCnt                              Acc72Ex[0].Data8[47661]
        #define CC3_bErrorPDOutCnt                             Acc72Ex[0].Data8[47662]
        #define CC3_bErrorSyncCnt                              Acc72Ex[0].Data8[47663]
        #define CC3_bSyncHskMode                               Acc72Ex[0].Data8[47664]
        #define CC3_bSyncSource                                Acc72Ex[0].Data8[47665]


#ifdef __PROFIBUS_DP_Master__
#define CC0_PD0_OUT_OFFSET_2BYTE        $980
#define CC0_PD0_OUT_SIZE_2BYTE          2880
#define CC0_PD0_IN_OFFSET_2BYTE         $14C0
#define CC0_PD0_IN_SIZE_2BYTE           2880
#define CC0_PD1_OUT_OFFSET_2BYTE        $8C0
#define CC0_PD1_OUT_SIZE_2BYTE          32
#define CC0_PD1_IN_OFFSET_2BYTE         $8E0
#define CC0_PD1_IN_SIZE_2BYTE           32
#endif

#ifdef __PROFIBUS_DP_Slave__
#define CC0_PD0_OUT_OFFSET_2BYTE        $980
#define CC0_PD0_OUT_SIZE_2BYTE          768
#define CC0_PD0_IN_OFFSET_2BYTE         $C80
#define CC0_PD0_IN_SIZE_2BYTE           768
#define CC0_PD1_OUT_OFFSET_2BYTE        $8C0
#define CC0_PD1_OUT_SIZE_2BYTE          32
#define CC0_PD1_IN_OFFSET_2BYTE         $8E0
#define CC0_PD1_IN_SIZE_2BYTE           32
#endif

#ifdef __DeviceNet_Master__
#define CC0_PD0_OUT_OFFSET_2BYTE        $980
#define CC0_PD0_OUT_SIZE_2BYTE          2880
#define CC0_PD0_IN_OFFSET_2BYTE         $14C0
#define CC0_PD0_IN_SIZE_2BYTE           2880
#define CC0_PD1_OUT_OFFSET_2BYTE        $8C0
#define CC0_PD1_OUT_SIZE_2BYTE          32
#define CC0_PD1_IN_OFFSET_2BYTE         $8E0
#define CC0_PD1_IN_SIZE_2BYTE           32
#endif

#ifdef __DeviceNet_Slave__
#define CC0_PD0_OUT_OFFSET_2BYTE        $980
#define CC0_PD0_OUT_SIZE_2BYTE          768
#define CC0_PD0_IN_OFFSET_2BYTE         $C80
#define CC0_PD0_IN_SIZE_2BYTE           768
#define CC0_PD1_OUT_OFFSET_2BYTE        $8C0
#define CC0_PD1_OUT_SIZE_2BYTE          32
#define CC0_PD1_IN_OFFSET_2BYTE         $8E0
#define CC0_PD1_IN_SIZE_2BYTE           32
#endif

#ifdef __CANopen_Master__
#define CC0_PD0_OUT_OFFSET_2BYTE        $980
#define CC0_PD0_OUT_SIZE_2BYTE          2880
#define CC0_PD0_IN_OFFSET_2BYTE         $14C0
#define CC0_PD0_IN_SIZE_2BYTE           2880
#define CC0_PD1_OUT_OFFSET_2BYTE        $8C0
#define CC0_PD1_OUT_SIZE_2BYTE          32
#define CC0_PD1_IN_OFFSET_2BYTE         $8E0
#define CC0_PD1_IN_SIZE_2BYTE           32
#endif

#ifdef __CANopen_Slave__
#define CC0_PD0_OUT_OFFSET_2BYTE        $980
#define CC0_PD0_OUT_SIZE_2BYTE          768
#define CC0_PD0_IN_OFFSET_2BYTE         $C80
#define CC0_PD0_IN_SIZE_2BYTE           768
#define CC0_PD1_OUT_OFFSET_2BYTE        $8C0
#define CC0_PD1_OUT_SIZE_2BYTE          32
#define CC0_PD1_IN_OFFSET_2BYTE         $8E0
#define CC0_PD1_IN_SIZE_2BYTE           32
```

```
#endif

#ifdef __CC_Link_Slave__
#define CC0_PD0_OUT_OFFSET_2BYTE       $980
#define CC0_PD0_OUT_SIZE_2BYTE         768
#define CC0_PD0_IN_OFFSET_2BYTE        $C80
#define CC0_PD0_IN_SIZE_2BYTE          768
#define CC0_PD1_OUT_OFFSET_2BYTE       $8C0
#define CC0_PD1_OUT_SIZE_2BYTE         32
#define CC0_PD1_IN_OFFSET_2BYTE        $8E0
#define CC0_PD1_IN_SIZE_2BYTE          32
#endif

#ifdef __EtherCAT_Master__
#define CC0_PD0_OUT_OFFSET_2BYTE       $980
#define CC0_PD0_OUT_SIZE_2BYTE         2880
#define CC0_PD0_IN_OFFSET_2BYTE        $14C0
#define CC0_PD0_IN_SIZE_2BYTE          2880
#define CC0_PD1_OUT_OFFSET_2BYTE       $8C0
#define CC0_PD1_OUT_SIZE_2BYTE         32
#define CC0_PD1_IN_OFFSET_2BYTE        $8E0
#define CC0_PD1_IN_SIZE_2BYTE          32
#endif

#ifdef __EtherCAT_Slave__
#define CC0_PD0_OUT_OFFSET_2BYTE       $980
#define CC0_PD0_OUT_SIZE_2BYTE         2880
#define CC0_PD0_IN_OFFSET_2BYTE        $14C0
#define CC0_PD0_IN_SIZE_2BYTE          2880
#define CC0_PD1_OUT_OFFSET_2BYTE       $8C0
#define CC0_PD1_OUT_SIZE_2BYTE         32
#define CC0_PD1_IN_OFFSET_2BYTE        $8E0
#define CC0_PD1_IN_SIZE_2BYTE          32
#endif

#ifdef __EtherNetIP_Scanner_Master__
#define CC0_PD0_OUT_OFFSET_2BYTE       $980
#define CC0_PD0_OUT_SIZE_2BYTE         2880
#define CC0_PD0_IN_OFFSET_2BYTE        $14C0
#define CC0_PD0_IN_SIZE_2BYTE          2880
#define CC0_PD1_OUT_OFFSET_2BYTE       $8C0
#define CC0_PD1_OUT_SIZE_2BYTE         32
#define CC0_PD1_IN_OFFSET_2BYTE        $8E0
#define CC0_PD1_IN_SIZE_2BYTE          32
#endif

#ifdef __EtherNetIP_Adapter_Slave__
#define CC0_PD0_OUT_OFFSET_2BYTE       $980
#define CC0_PD0_OUT_SIZE_2BYTE         2880
#define CC0_PD0_IN_OFFSET_2BYTE        $14C0
#define CC0_PD0_IN_SIZE_2BYTE          2880
#define CC0_PD1_OUT_OFFSET_2BYTE       $8C0
#define CC0_PD1_OUT_SIZE_2BYTE         32
#define CC0_PD1_IN_OFFSET_2BYTE        $8E0
#define CC0_PD1_IN_SIZE_2BYTE          32
#endif

#ifdef __Open_Modbus_TCP__
#define CC0_PD0_OUT_OFFSET_2BYTE       $980
#define CC0_PD0_OUT_SIZE_2BYTE         2880
#define CC0_PD0_IN_OFFSET_2BYTE        $14C0
#define CC0_PD0_IN_SIZE_2BYTE          2880
#define CC0_PD1_OUT_OFFSET_2BYTE       $8C0
#define CC0_PD1_OUT_SIZE_2BYTE         32
#define CC0_PD1_IN_OFFSET_2BYTE        $8E0
#define CC0_PD1_IN_SIZE_2BYTE          32
#endif

#ifdef __PROFINET_IO_Controller_Master__
#define CC0_PD0_OUT_OFFSET_2BYTE       $980
#define CC0_PD0_OUT_SIZE_2BYTE         2880
```

```
#define CC0_PD0_IN_OFFSET_2BYTE        $14C0
#define CC0_PD0_IN_SIZE_2BYTE          2880
#define CC0_PD1_OUT_OFFSET_2BYTE       $8C0
#define CC0_PD1_OUT_SIZE_2BYTE         32
#define CC0_PD1_IN_OFFSET_2BYTE        $8E0
#define CC0_PD1_IN_SIZE_2BYTE          32
#endif

#ifdef __PROFINET_IO_Device_Slave__
#define CC0_PD0_OUT_OFFSET_2BYTE       $980
#define CC0_PD0_OUT_SIZE_2BYTE         2880
#define CC0_PD0_IN_OFFSET_2BYTE        $14C0
#define CC0_PD0_IN_SIZE_2BYTE          2880
#define CC0_PD1_OUT_OFFSET_2BYTE       $8C0
#define CC0_PD1_OUT_SIZE_2BYTE         32
#define CC0_PD1_IN_OFFSET_2BYTE        $8E0
#define CC0_PD1_IN_SIZE_2BYTE          32
#endif


ptr CC0_PD0_OUT16(CC0_PD0_OUT_SIZE_2BYTE)->*;
ptr CC0_PD0_IN16(CC0_PD0_IN_SIZE_2BYTE)->*;
ptr CC0_PD1_OUT16(CC0_PD1_OUT_SIZE_2BYTE)->*;
ptr CC0_PD1_IN16(CC0_PD1_IN_SIZE_2BYTE)->*;
```

```
/****************************************************************************************************/
/* ACC-72EX requires different wait states compared to default wait state settings in Power PMAC
/* Write added wait state 60 nanoseconds
/* Delay of write line going low after chip select goes low 10 nanoseconds
/* Take write line high 20 nanosecond earlier FALSE
/* Read added wait state 60 nanoseconds
/* Delay of read line going low after chip select goes low 10 nanoseconds
/****************************************************************************************************/
Sys.BusCtrl[14]=$4646
Sys.BusCtrl[15]=$4646

global CommErrorFlag=0;
```

Initialization PLC

Recall that ACC-72EX requires a reset after each power up, power cycle, $$$ (reset), or $$$*** (factory default reset). This can be achieved with a startup (or initialization) PLC. Example:

```
// ACC-72EX initialization PLC
/****************************************/
open plc Acc72EX_StartupPLC
local endtime;
disable plc 2..31                              // Disable all other tasks

// Defining pointers for system channel mailboxes
L0=0
while(L0<84)
{
      CMD"SSMB_Data8(%d)->Acc72Ex[0].Data8[%d]",L0,L0+300
      sendallcmds
      L0++
}
L0=0
while(L0<42)
{
      CMD"SSMB_Data16(%d)->Acc72Ex[0].uData16[%d]",L0,L0+150
      sendallcmds
      L0++
}
L0=0
while(L0<21)
{
      CMD"SSMB_Data32(%d)->Acc72Ex[0].uData32[%d]",L0,L0+75
      sendallcmds
      L0++
}
L0=0
while(L0<84)
{
      CMD"SRMB_Data8(%d)->Acc72Ex[0].Data8[%d]",L0,L0+428
      sendallcmds
      L0++
}
L0=0
while(L0<42)
{
      CMD"SRMB_Data16(%d)->Acc72Ex[0].uData16[%d]",L0,L0+214
      sendallcmds
```

```
        L0++
}
L0=0
while(L0<21)
{
        CMD"SRMB_Data32(%d)->Acc72Ex[0].uData32[%d]",L0,L0+107
        sendallcmds
        L0++
}

// Defining pointers to Out/In PDOs

L0=0
while(L0<CC0_PD0_OUT_SIZE_2BYTE)
{
        CMD"CC0_PD0_OUT16(%d)-
>Acc72Ex[0].uData16[%d]",L0,L0+CC0_PD0_OUT_OFFSET_2BYTE;
        sendallcmds
        L0++
}
L0=0
while(L0< CC0_PD0_IN_SIZE_2BYTE)
{
        CMD"CC0_PD0_IN16(%d)-
>Acc72Ex[0].uData16[%d]",L0,L0+CC0_PD0_IN_OFFSET_2BYTE
        sendallcmds
        L0++
}
L0=0
while(L0<CC0_PD1_OUT_SIZE_2BYTE)
{
        CMD"CC0_PD1_OUT16(%d)-
>Acc72Ex[0].uData16[%d]",L0,L0+CC0_PD1_OUT_OFFSET_2BYTE
        sendallcmds
        L0++
}
L0=0
while(L0< CC0_PD1_IN_SIZE_2BYTE)
{
        CMD"CC0_PD1_IN16(%d)->Acc72Ex[0].uData16[%d]",L0,L0+
CC0_PD1_IN_OFFSET_2BYTE
        sendallcmds
        L0++
}
SCtrl_ulSystemCommandCOS=$55AA55AA        // Reset token for MASTER Unit
HCSC_HSF_RESET=1                    // Reset bit, token required for reset to
complete
CommErrorFlag=0;
endtime = Sys.Time + 2;                   // Reset Time-out Timer
while (CommErrorFlag==0 && HCSC_NSF_READY==0)    // Wait for reset to complete
{
        if (endtime<Sys.Time)                     // Check for reset timeout
        {
                CommErrorFlag = 1;
        }
}
call Timer(0.100); // 100 msec
```

```
// Toggle Communication Channel 0's Change of State Acknowledge bit in
// order to read the CC0_RCX_COMM_COS_RUN which is a part of Communication
// Channel 0 State Register
HCCC0_HCF_NETX_COS_ACK = HCCC0_HCF_NETX_COS_ACK ^ 1

if (CommErrorFlag==0)
{
      while ((CC0_RCX_COMM_COS_RUN) == 0)        // Wait for comm tasks to
      {
            // start on COMX modules
            // Repeating the toggle action for next while loop read
            HCCC0_HCF_NETX_COS_ACK = HCCC0_HCF_NETX_COS_ACK ^ 1
            call Timer(0.010); // 10 msec
      }
      enable plc Acc72EX_WatchdogPLC             // Enable the Watchdog plc
      enable plc Acc72EX_PDO_WritePLC            // Enable the write plc
      enable plc Acc72EX_PDO_ReadPLC             // Enable the read plc
}
disable plc Acc72EX_StartupPLC
close
```

The above PLC uses a Timer subprogram call that must be added to the PMAC Script Language→Libraries folder of the IDE project:

```
open subprog Timer(wait_duration) // wait_duration in seconds
local EndTime = Sys.Time + wait_duration;
while(Sys.Time < EndTime){}
close
```

Startup

To enable this startup PLC at power-up or reset, add the following line to pp_startup.txt in the Configuration folder:

```
enable plc Acc72EX_StartupPLC
```

Watchdog Function

The host Watchdog and the device Watchdog cells in the control block of each of the communication channels allow the operating system running on the netX to supervise the host or UMAC application and vice versa. There is no Watchdog function for the system block or for the handshake channel. The Watchdog for the channels is located in the control block of the status block of each communication channel.

The netX firmware reads the content of the device Watchdog cell, increments the value by one and copies it back into the host Watchdog location. Then, the application has to copy the new value from the host Watchdog location into the device Watchdog location. Copying the host Watchdog cell to the device Watchdog cell has to happen in the configured Watchdog time. When the overflow occurs, the firmware starts over and a one appears in the host Watchdog cell. A zero turns off the Watchdog and therefore never appears in the host Watchdog cell in the regular process.

The minimum Watchdog time is 20 ms. The application can start the Watchdog function by copying any value unequal to zero into device Watchdog cell. A zero in the device Watchdog location stops the Watchdog function. The Watchdog timeout is configurable in SYCON.net and downloaded to the netX firmware.

If the application fails to copy the value from the host Watchdog location to the device Watchdog location within the configured Watchdog time, the protocol stack will interrupt all network connections immediately, regardless of their current state. If the Watchdog tripped, then power cycling, channel reset, or channel initialization will allow the communication channel to open network connections again.

Here is sample code for copying the host Watchdog location to the device Watchdog location:

```
/****************************************/
// ACC-72EX Watchdog PLC
/****************************************/
open plc Acc72EX_WatchdogPLC
CC0_ulDeviceWatchdog = CC0_ulHostWatchdog // copies the host Watchdog content
                                    // to device Watchdog cell
                                    // for the ACC-72EX
close
```

Enabling the Communication Bus

Using the Bus On flag (CCx_RCX_APP_COS_BUS_ON, where x is the communication channel number), the host or UMAC application allows or disallows the netX firmware to open network connections. This flag is used together with the Bus On Enable flag (CCx_RCX_APP_COS_BUS_ON_ENABLE, where x is the communication channel number). If set, the netX firmware tries to open network connections; if cleared, no connections are allowed, and open connections are closed. If the Bus On Enable flag is set, it enables the execution of the Bus On command in the netX firmware.

```
CC0_RCX_APP_COS_BUS_ON=1            // Setting the Bus On flag for 1st ACC-72EX
CC0_RCX_APP_COS_BUS_ON_ENABLE=1     // Enabling the execution of Bus On Flag for 1st ACC-72EX

S_CC0_RCX_APP_COS_BUS_ON=1          // Setting the Bus On flag for 2nd ACC-72EX
S_CC0_RCX_APP_COS_BUS_ON_ENABLE=1   // Enabling the execution of Bus On Flag for 2nd ACC-72EX
```

Locating the Input/Output Data Image in PMAC

The header file provided for use with ACC-72EX provides proper addressing and offsets for each of the PDOs available for each communication module. There are also pointers declared in the header file and are defined as a part of the initialization PLC shown above. These pointers will be used to access different PDOs defined by SYCON.net software.

The following example PLCs are for reference only in order to demonstrate the proper handshaking necessary for reading and writing data to ACC-72EX from Power PMAC.

```
/****************************************/
// ACC-72EX Writing to PDO Sample PLC
/****************************************/
open plc Acc72EX_PDO_WritePLC
if (HCCC0_HCF_PD0_OUT_CMD == HCCC0_NCF_PD0_OUT_ACK)    {
// Making sure the ACK flag matches the CMD
// flag before writing the value to the
// output data image register
      P200=P200+1
      CC0_PD0_OUT16(0)= P200;         // write the output data image register
      // Toggle the CMD flag (^: XOR)
      HCCC0_HCF_PD0_OUT_CMD = HCCC0_HCF_PD0_OUT_CMD^1
      // indicating write completion
}
close
/****************************************/
```

```
// ACC-72EX Reading from PDO Sample PLC
/****************************************/

open plc Acc72EX_PDO_ReadPLC
if(HCCC0_NCF_PD0_IN_CMD == HCCC0_HCF_PD0_IN_ACK)
// If CMD flag and ACK flags are
// equal, then the input data image
// register can be read
{
      P201=CC0_PD0_IN16(0);   // read the input data image register
      HCCC0_HCF_PD0_IN_ACK = HCCC0_HCF_PD0_IN_ACK ^ 1
      // toggle the acknowledge bit
      // indicating read completion
}
close
```

DIAGNOSTICS

LEDs

There is one system LED (SYS LED) per ACC-72EX. SYS LED is always present as described below. There are up to 4 LEDs per communication and application channel. These LEDs, like the communication channel LED (COM LED), are network-specific and are described separately.

SYS LED

The system status LED (SYS LED) is always available. It indicates the state of the system and its protocol stacks. The following blink patterns are defined:

Color	State	Meaning
Yellow	Flashing Cyclically at 1 Hz	netX is in Boot Loader Mode and is Waiting for Firmware Download
	Solid	netX is in Boot Loader Mode, but an Error Occurred
Green	Solid	netX Operating System is Running and a Firmware is Started
Vellow / Green	Flashing Alternating	2nd Stage Bootloader is active
Off	N/A	netX has no Power Supply or Hardware Defect Detected

PROFIBUS-DP – Master – OPT10

The following table describes the meaning of the LEDs for the comX PROFIBUS-DP Master communication modules (COMX 100CA-DP/ COMX100CN-DP) when the firmware of the PROFIBUS DP Master protocol is loaded to the comX communication module:

COM LED (COM0)

Color	State	Meaning
Green	Flashing acyclic	No configuration or stack error
Green	Flashing cyclic	Profibus is configured, but bus communication is not yet released from the application
Green	On	Communication to all Slaves is established
Red	Flashing cyclic	Communication to at least one Slave is disconnected
Red	On	Communication to one/all Slaves is disconnected

PROFIBUS-DP – Slave – OPT11

The subsequent table describes the meaning of the LEDs for the comX PROFIBUS-DP Slave communication modules (COMX CA-DP/ COMX CNDP) when the firmware of the PROFIBUS DP Slave protocol is loaded to the comX communication module.

COM LED (COM0)

Color	State	Meaning
Green	On	RUN, cyclic communication
Red	Flashing cyclic	STOP, no communication, connection error
Red	Flashing acyclic	not configured

DeviceNet – Master – OPT20

The following table describes the meaning of the LEDs for the comX communication modules when the firmware of the DeviceNet Master protocol is loaded to the comX communication module:

MNS LED (COM0)

Color	State	Meaning
Green	On	Device is online and has established one or more connections
Green	Flashing	Device is online and has established no connection
Green/Red	Green/Red/Off	Self-test after power on: Green on for 0,25 s, then red on for 0,25 s, then off
Red	Flashing	Connection timeout
Red	On	Critical connection failure; device has detected a network error: duplicate MAC-ID or severe error in CAN network (CAN-bus off)
Red	Off	After start of the device and during duplicate MAC-ID check

DeviceNet – Slave – OPT21

The following table describes the meaning of the LEDs for the comX communication modules when the firmware of the DeviceNet Slave protocol is loaded to the comX communication module:

MNS LED (COM0)

Color	State	Meaning
Green	On	Device is online and has established one or more connections
Green	Flashing	Device is online and has established no connection
Green/Red	Green/Red/Off	Self-test after power on: Green on for 0,25 s, then red on for 0,25 s, then off
Red	Flashing	Connection timeout
Red	On	Critical connection failure; device has detected a network error: duplicate MAC-ID or severe error in CAN network (CAN-bus off)
Red	Off	After start of the device and during duplicate MAC-ID check

CANopen – Master – OPT30

The following table describes the meaning of the LEDs for the comX CANopen Master communication modules (COMX-CA-CO/ COMX-CNCOM) when the firmware of the CANopen Master protocol is loaded to the comX communication module:

CAN LED (COM0)

Color	State	Meaning
Green	Off	The device is executing a reset
Green	Single Flash	STOPPED: The Device is in STOPPED state
Green	Blinking	PREOPERATIONAL: The Device is in the PREOPERATIONAL state The indicator turns on and off with a frequency of 2,5 Hz: on for 200 ms, followed by off for 200 ms.
Green	On	OPERATIONAL: The Device is in the OPERATIONAL state
Red	Single flash	Warning Limit reached: At least one of the error counters of the CAN controller has reached or exceeded the warning level (too many error frames). The indicator shows one short flash (200 ms) followed by a long off phase (1,000 ms).
Red	Double flash	Error Control Event: A guard event (NMT Slave or NMTmaster) or a heartbeat event (Heartbeat consumer) has occurred. The indicator shows a sequence of two short flashes (each 200 ms), separated by a short off phase (200 ms). The sequence is finished by a long off phase (1,000 ms).
Red	On	Bus Off: The CAN controller is bus off

CANopen – Slave – OPT31

The following table describes the meaning of the LEDs for the comX CANopen Slave communication modules (COMX-CA-CO/ COMX-CNCOS) when the firmware of the CANopen Slave protocol is loaded to the comX communication module:

CAN LED (COM0)

Color	State	Meaning
Green	Off	The device is executing a reset
Green	Single Flash	STOPPED: The Device is in STOPPED state
Green	Blinking	PREOPERATIONAL: The Device is in the PREOPERATIONAL state The indicator turns on and off with a frequency of 2,5 Hz: on for 200 ms, followed by off for 200 ms.
Green	On	OPERATIONAL: The Device is in the OPERATIONAL state
Red	Off	No Error: The Device is in working condition
Red	Single flash	Warning Limit reached: At least one of the error counters of the CAN controller has reached or exceeded the warning level (too many error frames). The indicator shows one short flash (200 ms) followed by a long off phase (1,000 ms).
Red	Double flash	Error Control Event: A guard event (NMT Slave or NMTmaster) or a heartbeat event (Heartbeat consumer) has occurred. The indicator shows a sequence of two short flashes (each 200 ms), separated by a short off phase (200 ms). The sequence is finished by a long off phase (1,000 ms).
Red	On	Bus Off: The CAN controller is bus off

CC-Link – Slave – OPT51

The following table describes the meaning of the LEDs for the comX CCLink Slave communication modules (COMX 100CA-CCS/ COMX 100CNCCS) when the firmware of the CC-Link Slave protocol is loaded to the comX communication module:

RUN/ERR LED (COM0)

Color	State	Meaning
Green	Off	1. Before participating in the network 2. Unable to detect carrier 3. Timeout 4. Resetting hardware
Green	On	Receive both refresh and polling signals or just the refresh signal normally, after participating in the network.
Red	Off	1. Normal communication 2. Resetting hardware
Red	Blinking	The switch setting has been changed from the setting at the reset cancellation (blinks for 0.4 sec.).
Red	On	1. CRC error 2. Address parameter error (0, 65 or greater is set including the number of occupied stations) 3. Baud rate switch setting error during cancellation of reset (5 or greater)

EtherCAT – Master – OPT60

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX 100CA-RE/ COMX 100CN-RE) when the firmware of the EtherCAT Master protocol is loaded to the comX communication module:

RUN LED (COM0)

Color	State	Meaning
Green	Off	INIT: The device is in state INIT
Green	Blinking	PRE-OPERATIONAL: The device is in PREOPERATIONAL state
Green	Flickering	BOOT: Device is in BOOT state
Green	Single Flash	SAFE-OPERATIONAL: The device is in SAFE-OPERATIONAL state
Green	On	OPERATIONAL: The device is in OPERATIONAL state

ERR LED (COM1)

Color	State	Meaning
Red	Off	Master has no errors
Red	On	Master has detected a communication error. The error is indicated in the DPM

LINK LED

Green LED on ETH0 connector

Color	State	Meaning
Green	On	A link is established
Green	Off	No link established

ACT LED

Yellow LED on ETH0 connector

Color	State	Meaning
Yellow	Flashing	The device sends/receives Ethernet frames

LED State Definition for EtherCAT Master for the RUN and ERR LEDs

Color	Meaning
On	The indicator is constantly on.
Off	The indicator is constantly off.
Blinking	The indicator turns on and off with a frequency of 2,5 Hz: on for 200 ms, followed by off for 200 ms.
Flickering	The indicator turns on and off with a frequency of approximately 10 Hz: on for approximately 50 ms, followed by off for 50 ms.
Single Flash	The indicator shows one short flash (200 ms) followed by a long off phase (1,000 ms).
Double Flash	The indicator shows a sequence of two short flashes (each 200 ms), separated by a short off phase (200 ms). The sequence is finished by a long off phase (1,000 ms).

EtherCAT – Slave – OPT61

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX 100CA-RE/ COMX 100CN-RE) when the firmware of the EtherCAT Slave protocol is loaded to the comX communication module:

RUN LED (COM0)

Color	State	Meaning
Green	Off	INIT: The device is in state INIT
Green	Blinking	PRE-OPERATIONAL: The device is in PREOPERATIONAL state
Green	Flickering	BOOT: Device is in BOOT state
Green	Single Flash	SAFE-OPERATIONAL: The device is in SAFE-OPERATIONAL state
Green	On	OPERATIONAL: The device is in OPERATIONAL state

ERR LED (COM1)

Color	State	Meaning
Red	Off	No error: The EtherCAT communication of the device is in working condition
Red	Blinking	Invalid Configuration: General Configuration Error Possible reason: State change commanded by master is impossible due to register or object settings.
Red	Single Flash	Local Error: Slave device application has changed the EtherCAT state autonomously. Possible reason 1: A host Watchdog timeout has occurred. Possible reason 2: Synchronization Error, device enters Safe-Operational automatically.
Red	Double Flash	Application Watchdog Timeout: An application Watchdog timeout has occurred. Possible reason: Sync Manager Watchdog timeout.

LINK/ACT LED

Green LED on ETH0(IN) / ETH1(OUT) connectors:

Color	State	Meaning
Green	On	A link is established
Green	Flashing	The device sends/receives Ethernet frames
Green	Off	No link established

Yellow LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Yellow	-	-

LED State Definition for EtherCAT Slave for the RUN and ERR LEDs

Color	Meaning
On	The indicator is constantly on.
Off	The indicator is constantly off.
Blinking	The indicator turns on and off with a frequency of 2,5 Hz: on for 200 ms, followed by off for 200 ms.
Single Flash	The indicator shows one short flash (200 ms) followed by a long off phase (1,000 ms).
Double Flash	The indicator shows a sequence of two short flashes (each 200 ms), separated by a short off phase (200 ms). The sequence is finished by a long off phase (1,000 ms).

EtherNet/IP – Scanner/Master – OPT70

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX-CA-RE/ COMX-CNRE) when the firmware of the EtherNet/IP Scanner (Master) protocol is loaded to the comX communication module:

MS LED (COM0)

Color	State	Meaning
Green	On	Device operational: If the device is operating correctly, the module status indicator shall be steady green.
Green	Flashing	Standby: If the device has not been configured, the module status indicator shall be flashing green.
Red	On	Major fault: If the device has detected a non-recoverable major fault, the module status indicator shall be steady red.
Green	Flashing	Minor fault: If the device has detected a recoverable minor fault, the module status indicator shall be flashing red. NOTE: An incorrect or inconsistent configuration would be considered a minor fault.
Red/Green	Flashing	Self-test: While the device is performing its power up testing, the module status indicator shall be flashing green/red.
-	Off	No power: If no power is supplied to the device, the module status indicator shall be steady off.

NS LED (COM1)

Color	State	Meaning
Green	On	Connected: If the device has at least one established connection (even to the Message Router), the network status indicator shall be steady green.
Green	Flashing	No connections: If the device has no established connections, but has obtained an IP address, the network status indicator shall be flashing green.
Red	On	Duplicate IP: If the device has detected that its IP address is already in use, the network status indicator shall be steady red.
Red	Flashing	Connection timeout: If one or more of the connections in which this device is the target has timed out, the network status indicator shall be flashing red. This shall be left only if all timed out connections are reestablished or if the device is reset.
Red/Green	Flashing	Self-test: While the device is performing its power up testing, the network status indicator shall be flashing green/red.
-	Off	Not powered, no IP address: If the device does not have an IP address (or is powered off), the network status indicator shall be steady off.

LINK LED

Green LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Green	On	A connection to the Ethernet exists
Green	Off	The device has no connection to the Ethernet

ACT LED

Yellow LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Yellow	Flashing	The device sends/receives Ethernet frames

EtherNet/IP – Adaptor/Slave – OPT71

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX-CA-RE/ COMX-CNRE) when the firmware of the EtherNet/IP Adapter (Slave) protocol is loaded to the comX communication module:

MS LED (COM0)

Color	State	Meaning
Green	On	Device operational: If the device is operating correctly, the module status indicator shall be steady green.
Green	Flashing	Standby: If the device has not been configured, the module status indicator shall be flashing green.
Red	On	Major fault: If the device has detected a non-recoverable major fault, the module status indicator shall be steady red.
Green	Flashing	Minor fault: If the device has detected a recoverable minor fault, the module status indicator shall be flashing red. NOTE: An incorrect or inconsistent configuration would be considered a minor fault.
Red/Green	Flashing	Self-test: While the device is performing its power up testing, the module status indicator shall be flashing green/red.
-	Off	No power: If no power is supplied to the device, the module status indicator shall be steady off.

NS LED (COM1)

Color	State	Meaning
Green	On	Connected: If the device has at least one established connection (even to the Message Router), the network status indicator shall be steady green.
Green	Flashing	No connections: If the device has no established connections, but has obtained an IP address, the network status indicator shall be flashing green.
Red	On	Duplicate IP: If the device has detected that its IP address is already in use, the network status indicator shall be steady red.
Red	Flashing	Connection timeout: If one or more of the connections in which this device is the target has timed out, the network status indicator shall be flashing red. This shall be left only if all timed out connections are reestablished or if the device is reset.
Red/Green	Flashing	Self-test: While the device is performing its power up testing, the network status indicator shall be flashing green/red.
-	Off	Not powered, no IP address: If the device does not have an IP address (or is powered off), the network status indicator shall be steady off.

LINK LED

Green LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Green	On	A connection to the Ethernet exists
Green	Off	The device has no connection to the Ethernet

ACT LED

Yellow LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Yellow	Flashing	The device sends/receives Ethernet frames

Open Modbus/TCP – OPT80

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX 100CA-RE/ COMX 100CN-RE) when the firmware of the Open Modbus/TCP protocol is loaded to the comX communication module:

RUN LED (COM0)

Color	State	Meaning
Green	Off	Not Ready OMB task is not ready
Green	Flashing cyclic with 1Hz	Ready, not configured yet OMB task is ready and not configured yet
Green	Flashing cyclic with 5Hz	Waiting for Communication: OMB task is configured
Green	On	Connected: OMB task has communication – at least one TCP connection is established

ERR LED (COM1)

Color	State	Meaning
Red	Off	No communication error
Red	Flashing cyclic with 2Hz (On/Off ratio 25%)	System error
Red/Green	On	Communication error active

LINK LED

Green LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Green	On	A connection to the Ethernet exists
Green	Off	The device has no connection to the Ethernet

ACT LED

Yellow LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Yellow	Flashing	The device sends/receives Ethernet frames

PROFINET IO – Controller – OPT90

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX 100CA-RE/ COMX 100CN-RE) when the firmware of the PROFINET IO-RT Controller protocol is loaded to the comX communication module:

SF LED (COM0)

Color	State	Meaning
Red	On	(together with BF „red ON“) No valid Master license
Red	Flashing cyclic with 2Hz	System error: Invalid configuration, Watchdog error or internal error
Red	Off	No error

BF LED (COM1)

Color	State	Meaning
Red	On	No Connection: No Link or (together with SF „red ON“) No valid Master license
Red	Flashing cyclic with 2Hz	Configuration fault: not all configured IO-Devices are connected.
Red	Off	No error

LINK LED

Green LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Green	On	A connection to the Ethernet exists
Green	Off	The device has no connection to the Ethernet

ACT LED

Yellow LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Yellow	Flashing	The device sends/receives Ethernet frames

PROFINET IO – Device – OPT91

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX-CA-RE/ COMX-CNRE) when the firmware of the PROFINET IO-RT-Device protocol is loaded to the comX communication module:

SF LED (COM0)

Color	State	Meaning
Red	On	Watchdog timeout; channel, generic or extended diagnosis present; system error
Red	Flashing cyclic with 2Hz (for 3 seconds)	DCP signal service is initiated via the bus
Red	Off	No error

BF LED (COM1)

Color	State	Meaning
Red	On	No configuration; or low speed physical link; or no physical link
Red	Flashing cyclic with 2Hz	No data exchange
Red	Off	No error

LINK LED

Green LED on ETH0 / ETH1 connectors

Color	State	Meaning
Green	On	A connection to the Ethernet exists
Green	Off	The device has no connection to the Ethernet

ACT LED

Yellow LED on ETH0 / ETH1 connectors

Color	State	Meaning
Yellow	Flashing	The device sends/receives Ethernet frames

APPENDIX A – SETUP EXAMPLES

SYCON.net Setup

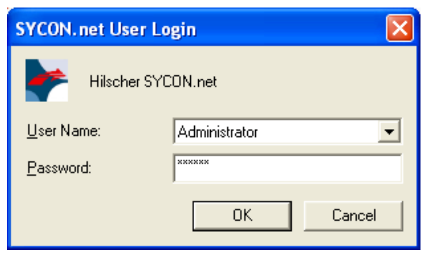
The following is a sample setup using an ACC-72EX Ethernet IP slave with an Allen-Bradley CompactLogix controller (1769-L18ERM-BB1B) as a master. SYCON.net for netX 1.310 was used in this example.

With the power off, plug the ACC-72EX into the UBUS backplane, and then power the UMAC rack. Connect the diagnostic port to a USB port on the PC using a micro-USB type cable.

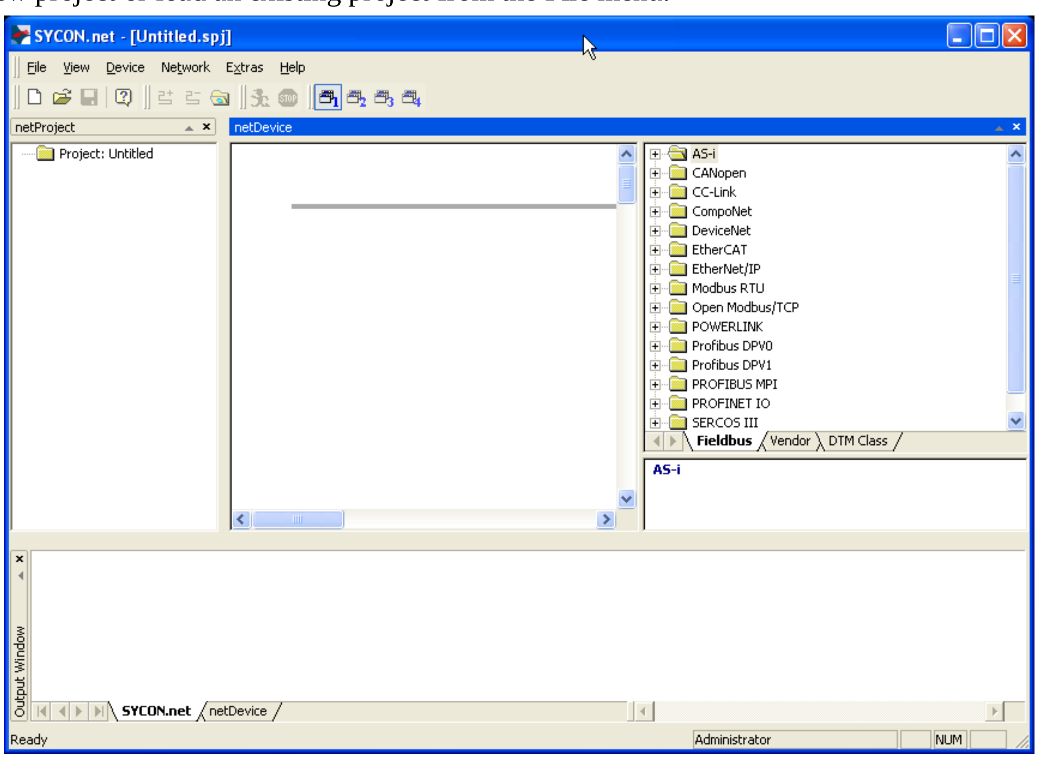
Launch the SYCON.NET software on the PC.



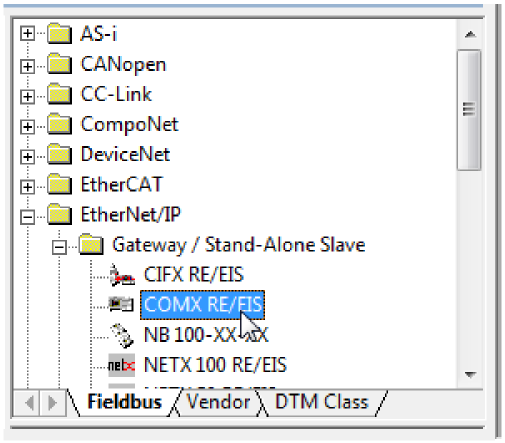
Enter the password:



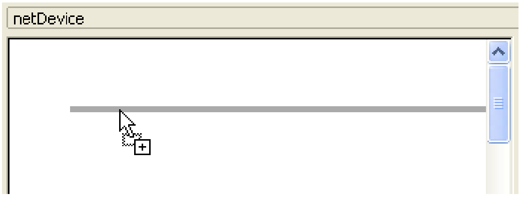
Start a new project or load an existing project from the File menu:

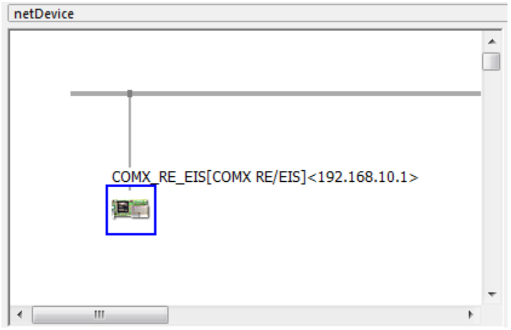


Select the COMX module, to which the USB is connected, from the Fieldbus protocol list:

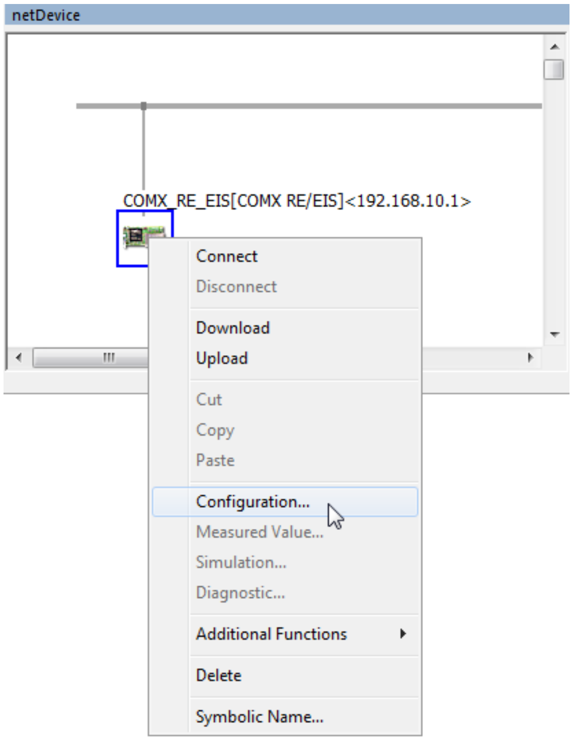


Drag and drop the module onto the BusLine in the netDevice window (notice that the module can only be inserted on the BusLine):

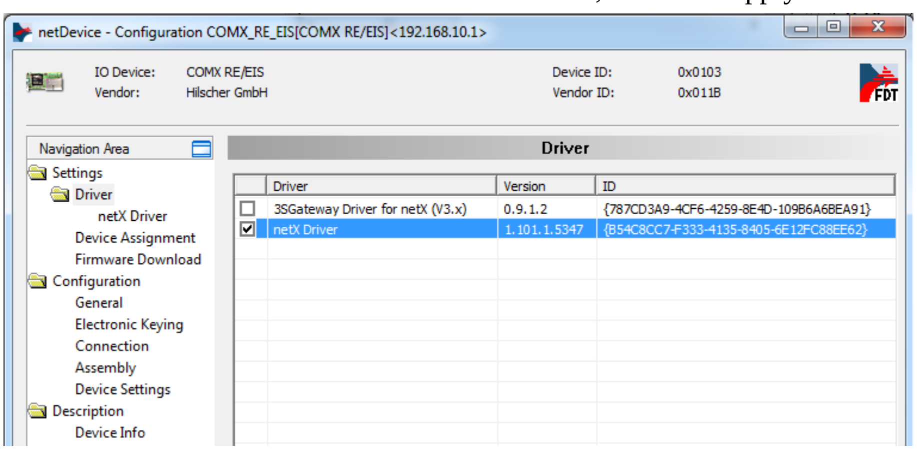




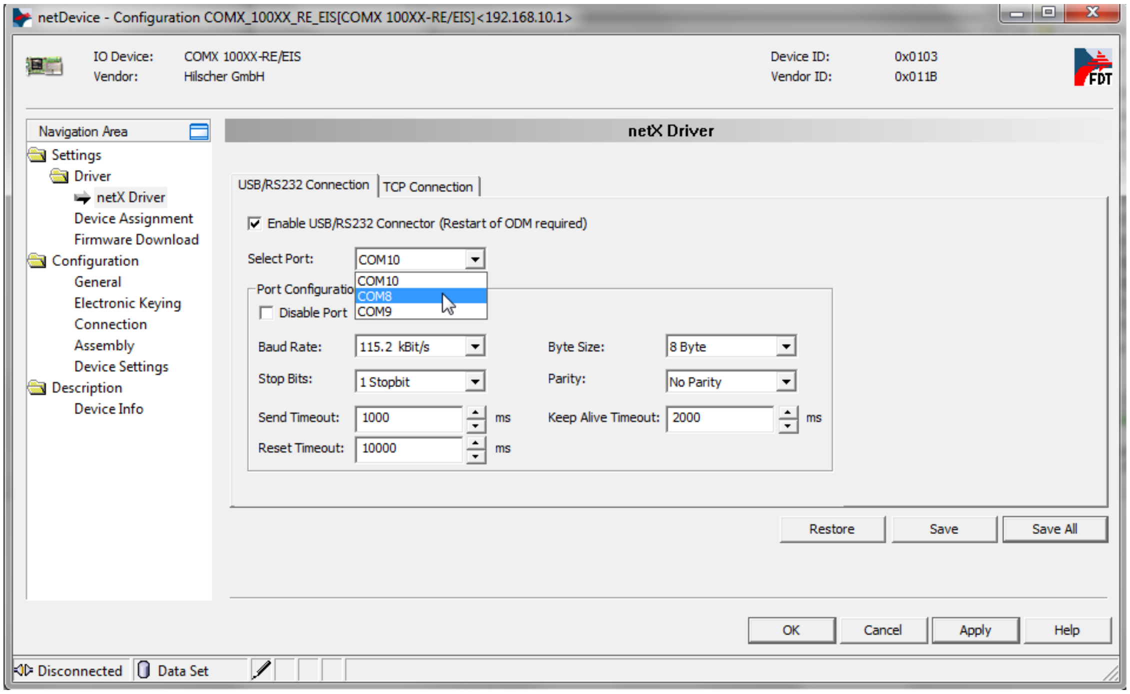
Establish USB communications to the COMX gateway by right clicking on the device icon and selecting "Configuration…":



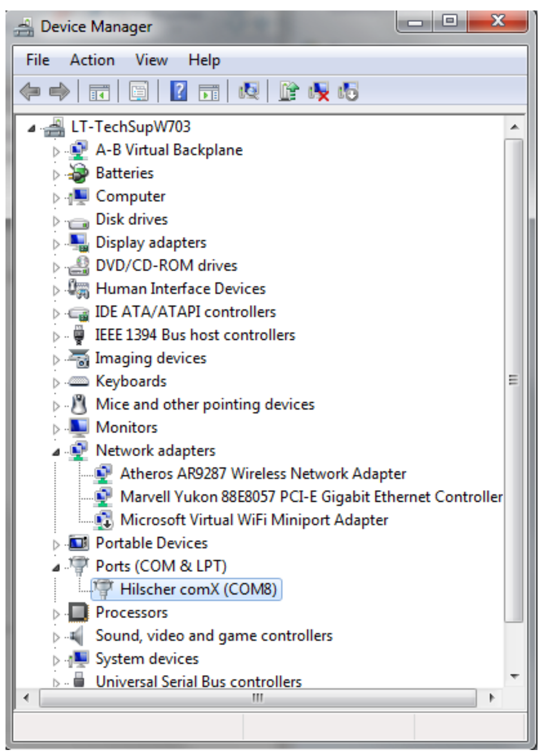
In the netDevice Configuration window, select the Driver folder under Settings folder in the Navigation Area, check the checkmark box for netX Driver on the driver list, and click Apply:



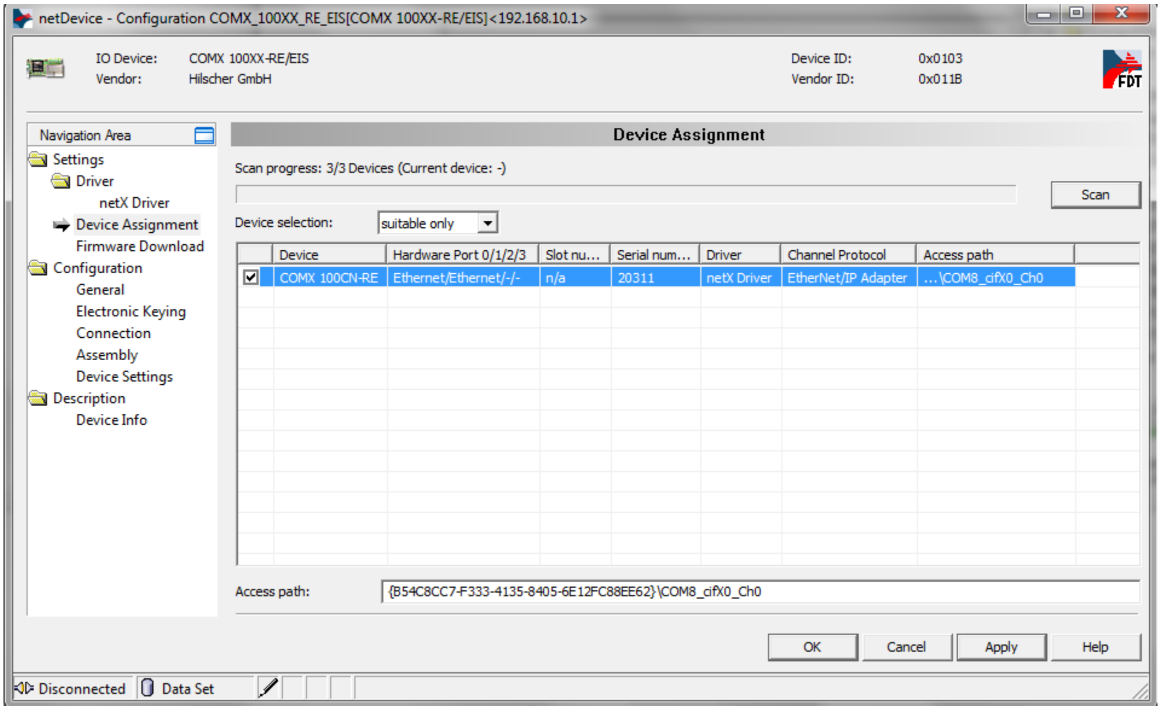
Select the netX Driver node in Driver folder in the Navigation Area, and select the port for the USB connection to the COMX module. Click Save and Apply (just click OK if Apply is grayed out):



Note: You can Check Windows Device Manager in order to identify which COM port provides the connection to the Hilscher COMX module:



Click the Device Assignment under Driver folder in the Navigation Area. Assign the netX Driver to the detected COMX module by checking the checkmark box next to the detected device, and click Apply:



When used with Turbo PMAC, the reset line is released too fast for some Hilscher COMX modules, which puts them in a boot mode. This can prevent the device from being detected by Sycon.NET software. Make sure the device receives a system wide reset using the PMAC suggested M-variables ulSystemCommandCOS and HSF_RESET registers as shown here.

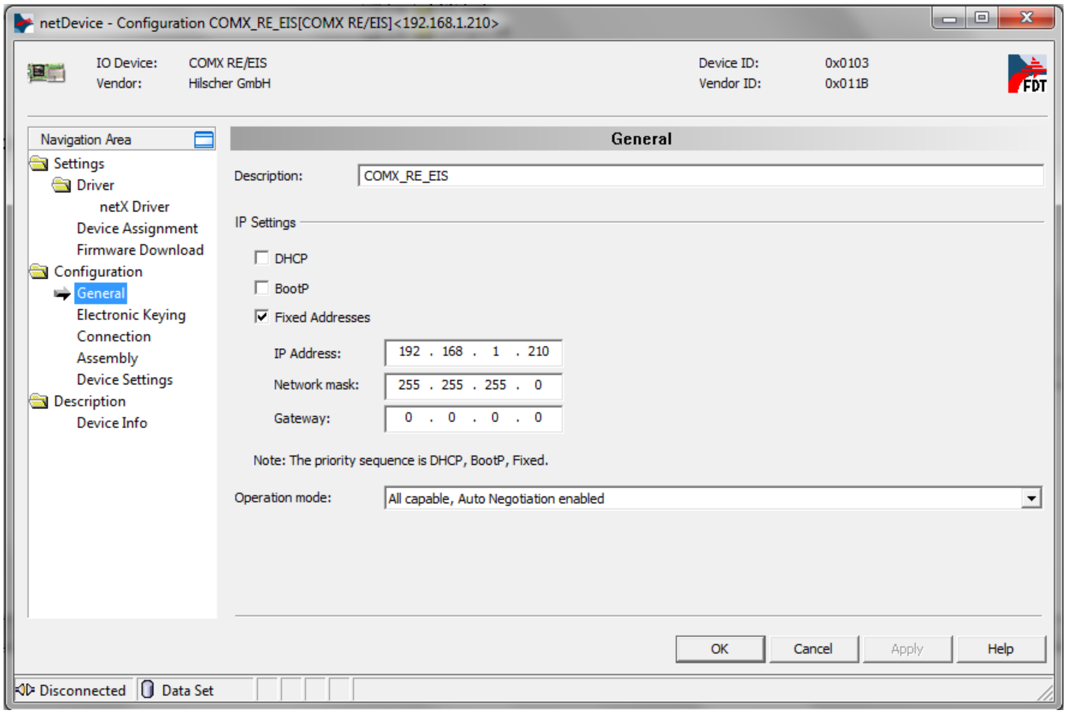
Note

SCtrl_ulSystemCommandCOS=$55AA55AA

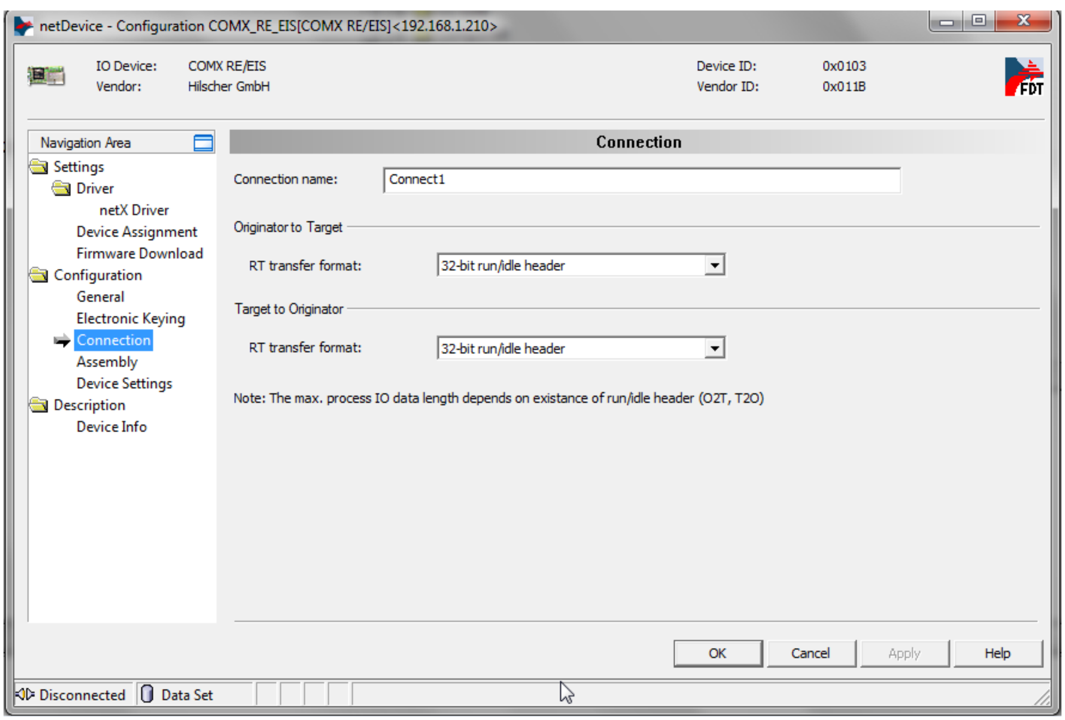
HCSC_HSF_RESET=1

Note that ACC-72EX Setup Assistant software automatically resets the cards if it cannot detect the identification cookie.

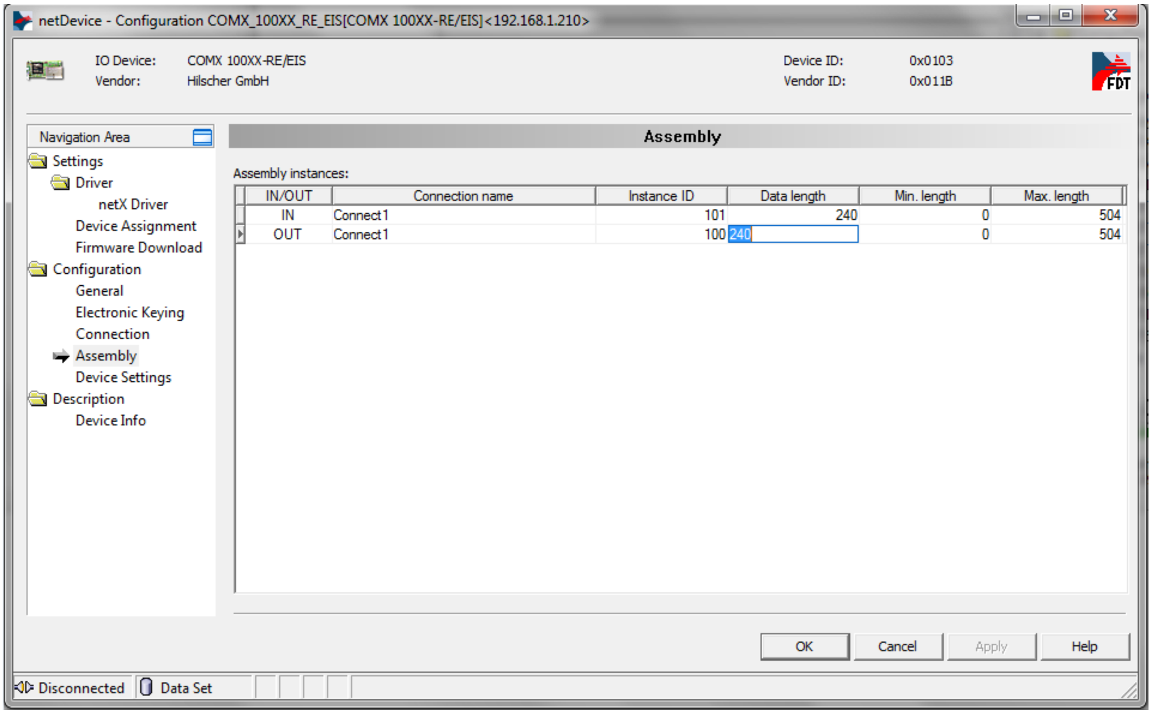
Set the IP address for the COMX module in the General Configuration window:



Set Connections:



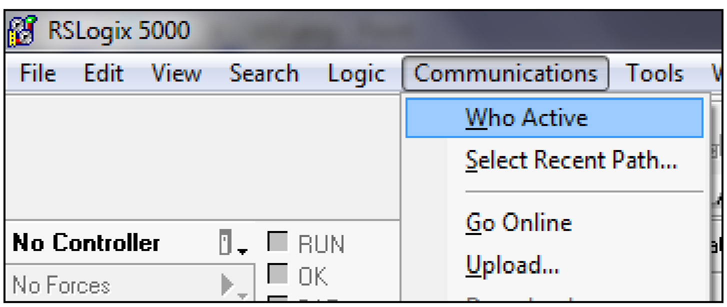
Set Instance IDs and Data lengths in the Assembly window. 240 is the maximum length for the CompactLogix 1769-L18ERM-BB1B controller.



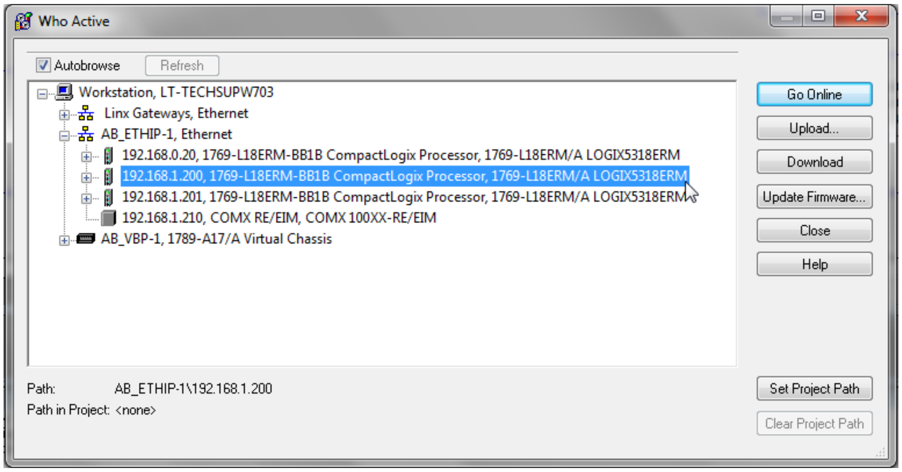
RSLogix 5000 Setup

RSLogix 5000 version 20 is used in this example.

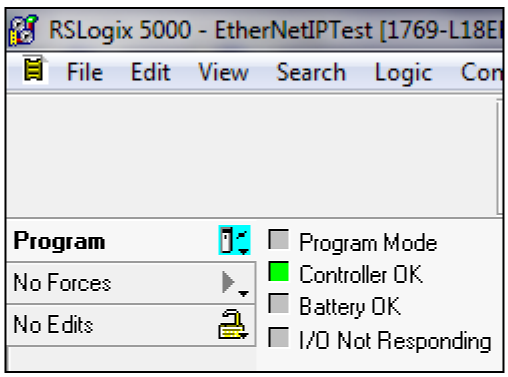
Launch RSLogix, and click on Who Active in the Communications pull down menu to find the CompactLogix controller:



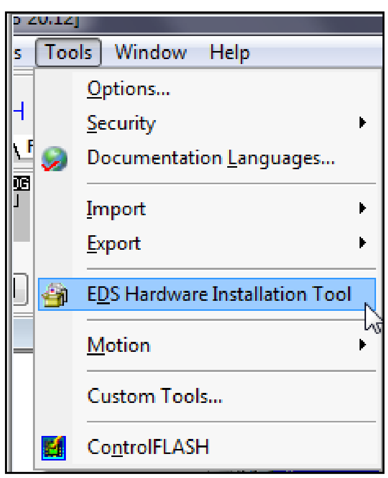
Select the controller, and click the Go Online button to test communication:



The Controller OK indicator box should change to green like below:



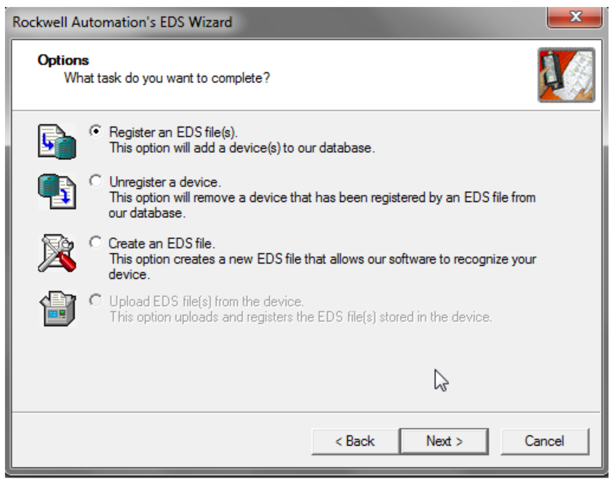
Next, install the EDS file for the Hilscher COMX slave of the ACC-72EX. Go to the Tools pull down menu, and select EDS Hardware Installation Tool:



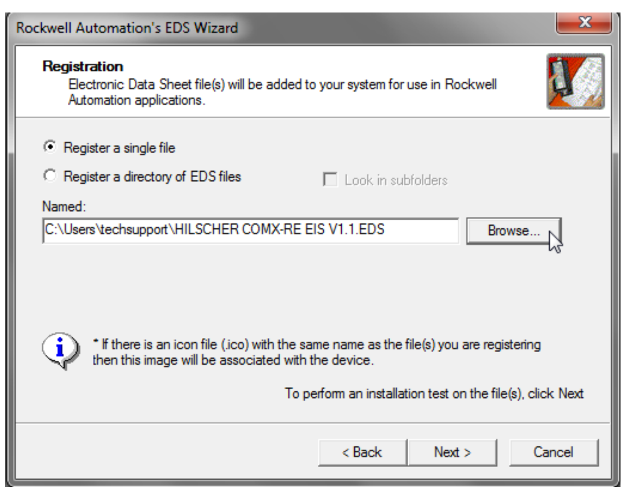
Click Next:



Select the Register an EDS file(s) radial button:

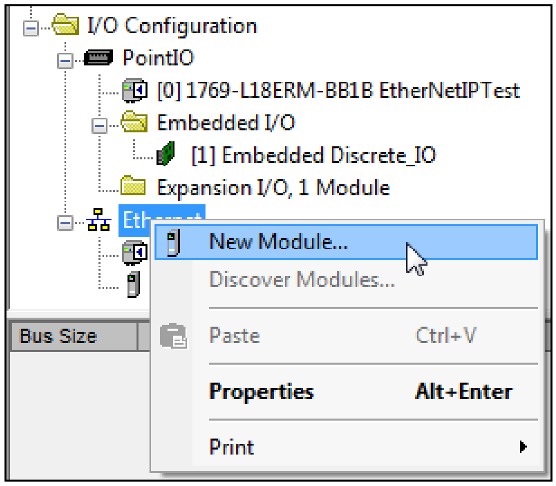


Browse to and select the Hilsher EDS file. EDS files can be downloaded at hilscher.com at http://www.hilscher.com/hcuk/support_software.html . Click Next.

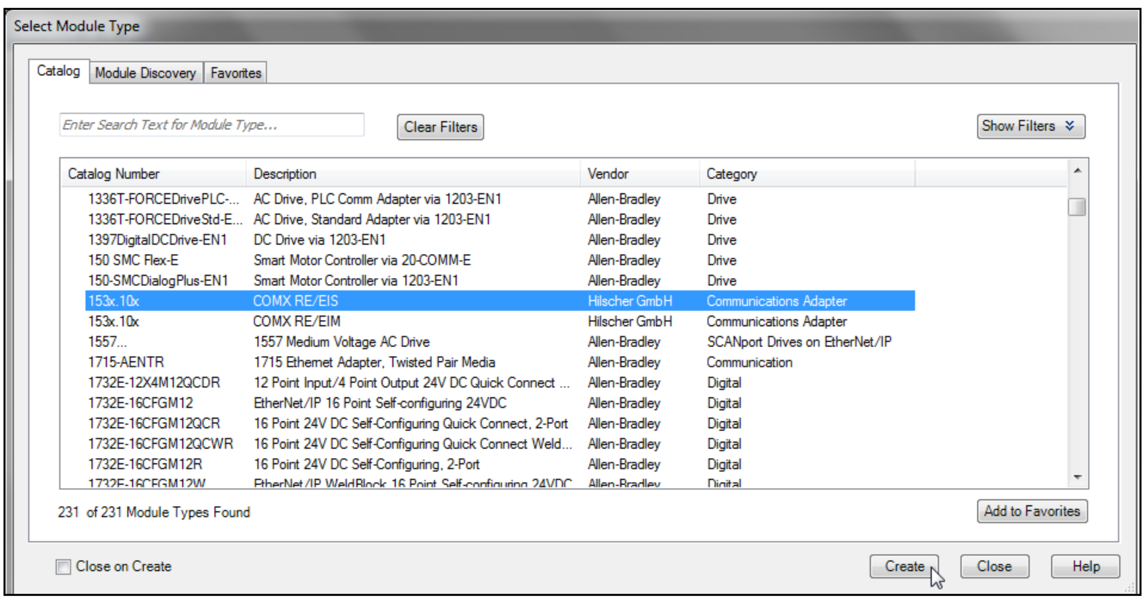


Follow the directions in the remaining windows for finishing the EDS installation.

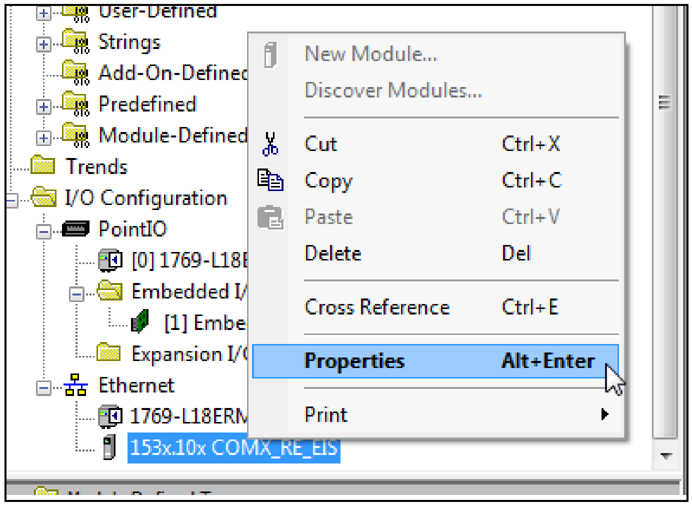
The next step will apply the EDS installation, but first the controller needs to be offline. Click the Go Offline selection under the Communications tab (Go Offline is displayed when the controller is online, and Go Online is displayed when offline). Under I/O Configuration in the Controller Organizer, right-click on Ethernet, and select New Module…:



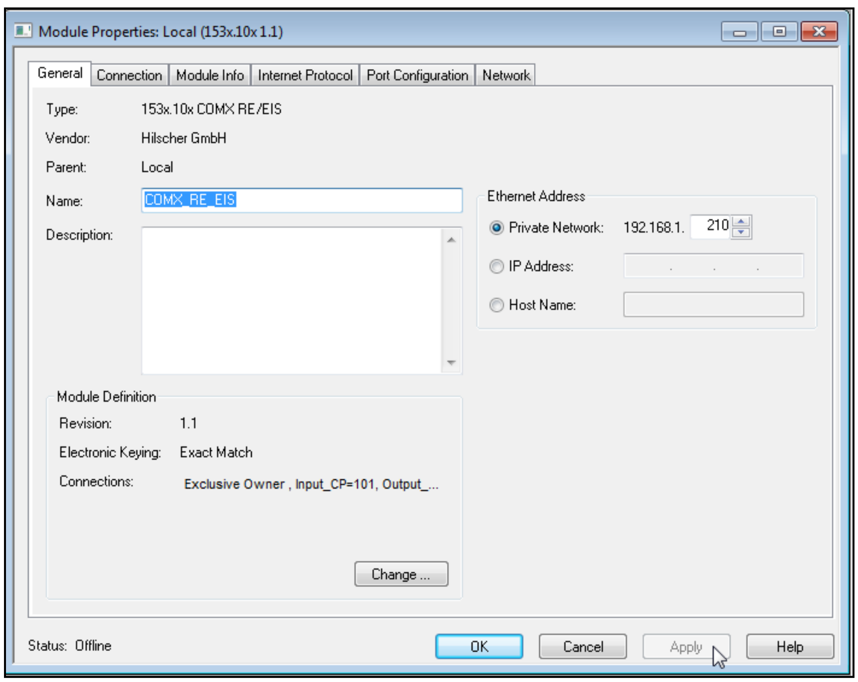
Scroll down to and select the COMX slave module, and press Create:



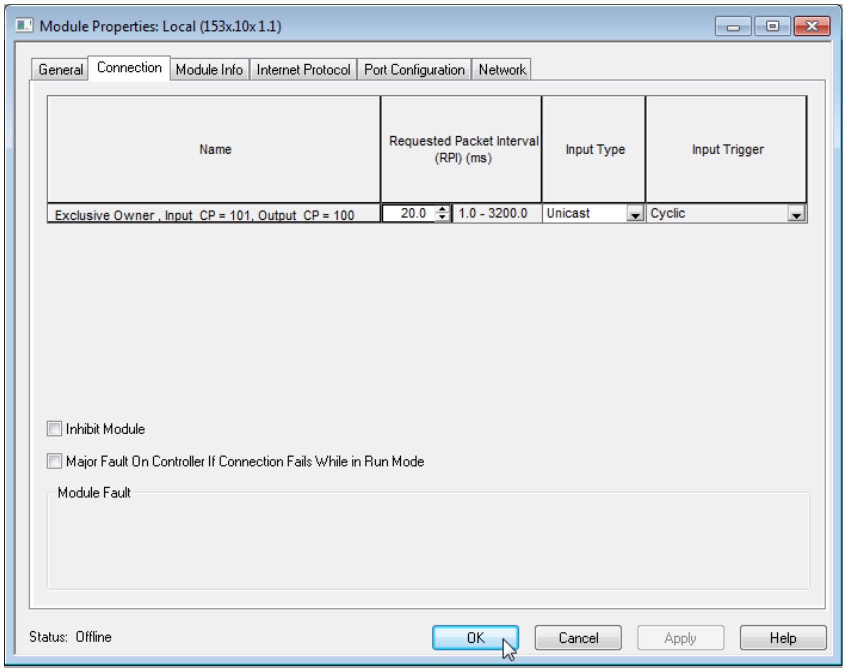
The created entry should appear under Ethernet in the Control Organizer. Right-click on it, and select Properties:



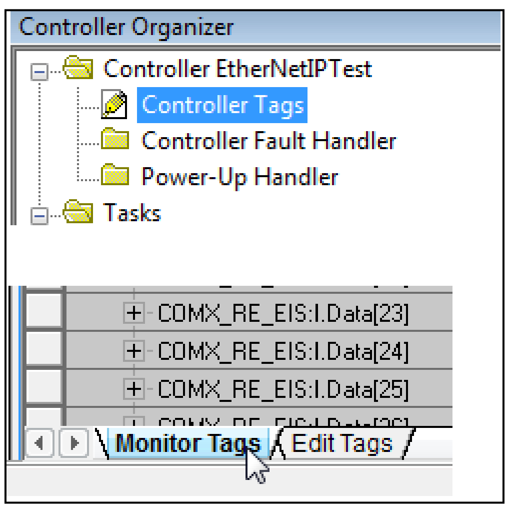
Under the General tab, set the IP address of the ACC-72EX:



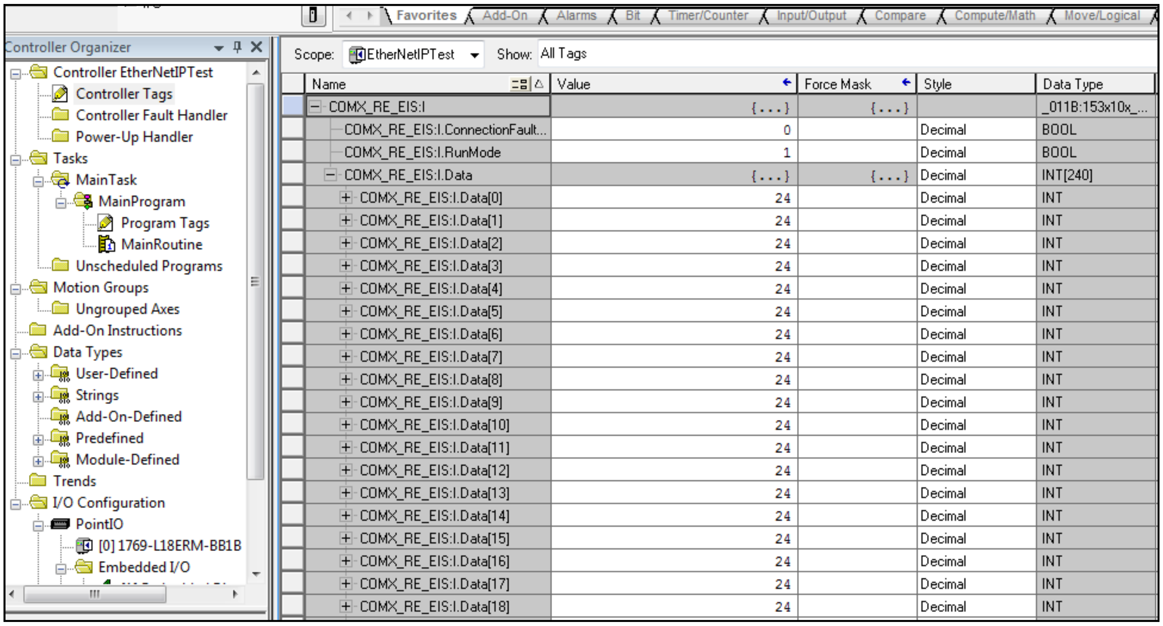
Check the settings under the Connection tab:



Double-click on Controller Tags, and open the Monitor Tags tab:

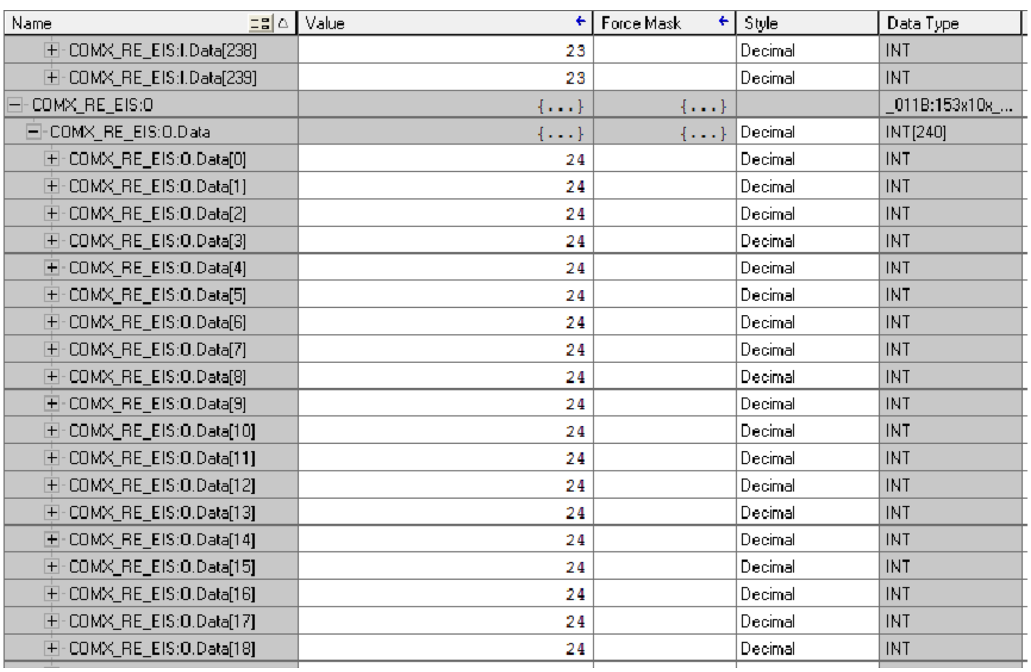


Click on "+" to expand the input data entries. Now input values from the ACC-72EX can be seen in the Value column (controller must be online).



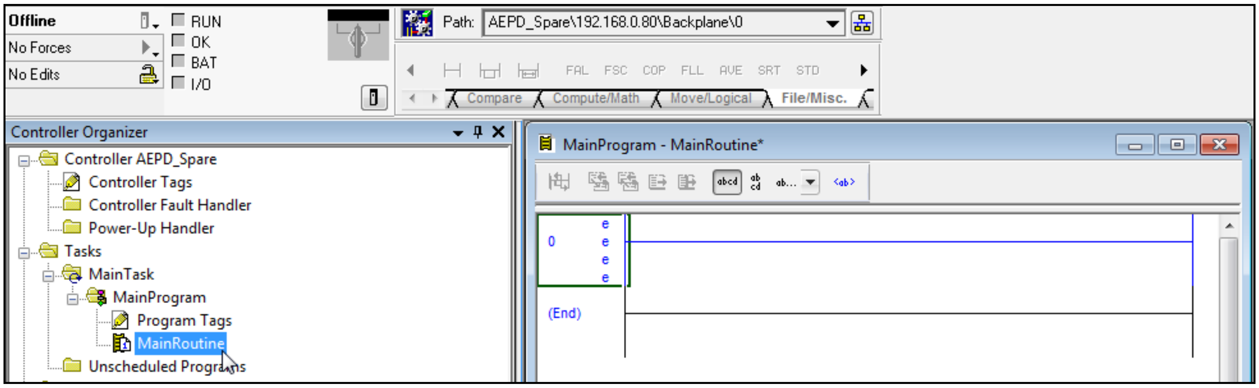
Name	Value	Force Mask	Style	Data Type
COMX_RE_EIS:I	{...}	{...}		_011B:153x10x_...
COMX_RE_EIS:I.ConnectionFault...	0		Decimal	BOOL
COMX_RE_EIS:I.RunMode	1		Decimal	BOOL
COMX_RE_EIS:I.Data	{...}	{...}	Decimal	INT[240]
COMX_RE_EIS:I.Data[0]	24		Decimal	INT
COMX_RE_EIS:I.Data[1]	24		Decimal	INT
COMX_RE_EIS:I.Data[2]	24		Decimal	INT
COMX_RE_EIS:I.Data[3]	24		Decimal	INT
COMX_RE_EIS:I.Data[4]	24		Decimal	INT
COMX_RE_EIS:I.Data[5]	24		Decimal	INT
COMX_RE_EIS:I.Data[6]	24		Decimal	INT
COMX_RE_EIS:I.Data[7]	24		Decimal	INT
COMX_RE_EIS:I.Data[8]	24		Decimal	INT
COMX_RE_EIS:I.Data[9]	24		Decimal	INT
COMX_RE_EIS:I.Data[10]	24		Decimal	INT
COMX_RE_EIS:I.Data[11]	24		Decimal	INT
COMX_RE_EIS:I.Data[12]	24		Decimal	INT
COMX_RE_EIS:I.Data[13]	24		Decimal	INT
COMX_RE_EIS:I.Data[14]	24		Decimal	INT
COMX_RE_EIS:I.Data[15]	24		Decimal	INT
COMX_RE_EIS:I.Data[16]	24		Decimal	INT
COMX_RE_EIS:I.Data[17]	24		Decimal	INT
COMX_RE_EIS:I.Data[18]	24		Decimal	INT

Click on "+" to expand the output data entries. The values seen in the Value column should now be seen as inputs in the ACC-72EX. Values can be changed here manually, or in program logic such as in the ladder logic example that follows.

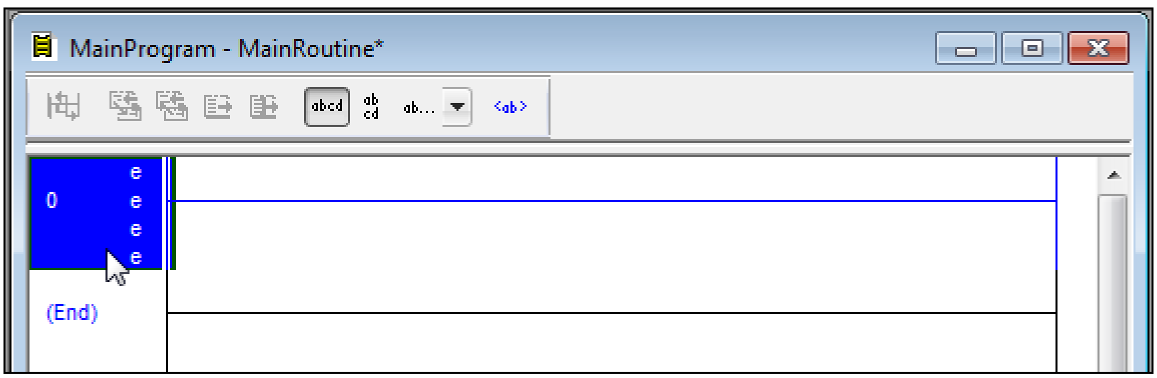


Name	Value	Force Mask	Style	Data Type
COMX_RE_EIS:I.Data[238]	23		Decimal	INT
COMX_RE_EIS:I.Data[239]	23		Decimal	INT
COMX_RE_EIS:O	{...}	{...}		_011B:153x10x_...
COMX_RE_EIS:O.Data	{...}	{...}	Decimal	INT[240]
COMX_RE_EIS:O.Data[0]	24		Decimal	INT
COMX_RE_EIS:O.Data[1]	24		Decimal	INT
COMX_RE_EIS:O.Data[2]	24		Decimal	INT
COMX_RE_EIS:O.Data[3]	24		Decimal	INT
COMX_RE_EIS:O.Data[4]	24		Decimal	INT
COMX_RE_EIS:O.Data[5]	24		Decimal	INT
COMX_RE_EIS:O.Data[6]	24		Decimal	INT
COMX_RE_EIS:O.Data[7]	24		Decimal	INT
COMX_RE_EIS:O.Data[8]	24		Decimal	INT
COMX_RE_EIS:O.Data[9]	24		Decimal	INT
COMX_RE_EIS:O.Data[10]	24		Decimal	INT
COMX_RE_EIS:O.Data[11]	24		Decimal	INT
COMX_RE_EIS:O.Data[12]	24		Decimal	INT
COMX_RE_EIS:O.Data[13]	24		Decimal	INT
COMX_RE_EIS:O.Data[14]	24		Decimal	INT
COMX_RE_EIS:O.Data[15]	24		Decimal	INT
COMX_RE_EIS:O.Data[16]	24		Decimal	INT
COMX_RE_EIS:O.Data[17]	24		Decimal	INT
COMX_RE_EIS:O.Data[18]	24		Decimal	INT

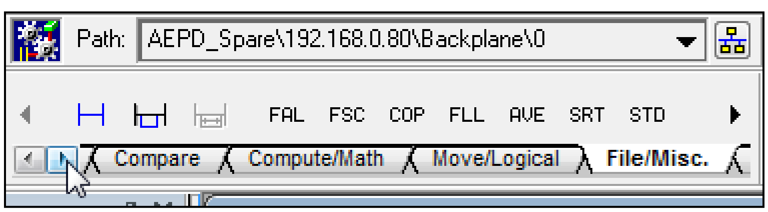
The following is an example which uses a "Copy" function to transfer all of the input values into corresponding output values. Double-click on MainRoutine:



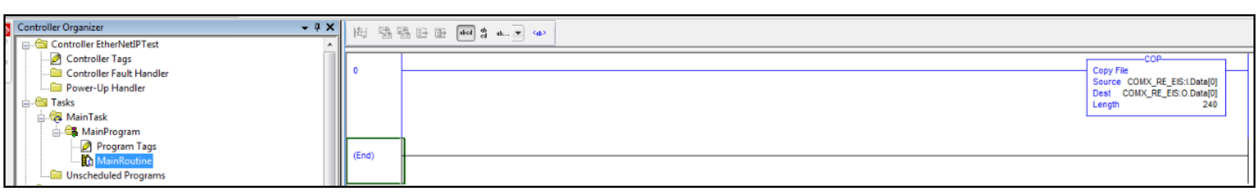
Click to select the top rung:



Click on the right arrow as needed to bring into view the File/Misc. ladder entries tab:



Click on the File/Misc. tab, and then drag and drop COP (copy) onto a rung. Look for a green dot to appear on the left side of the rung when the cursor is hovered there, and then drop the COP function. To copy all the ACC-72EX inputs into corresponding CopactLogix outputs, set Source to COMX_RE_EIS:I.Data[0], Dest to COMX_RE_EIS:O.Data[0], and Length to 240:



COMX Test PLC

The following Turbo PMAC PLCs can be used to test the communication of the COMX module. Run PLC 1 and check the variable M_CommErrorFlag. If it is =0 after PLC 1 finishes, the COMX module is communicating properly.

```
CLOSE
END GAT
DEL GAT
#include "M-VariableDefinition_$6C000.pmc"


#define M_CommErrorFlag P1
#define timer i6612
#define msec *8388608/i10while(i6612>0)endwhile

OPEN PLC 1 CLEAR
DISABLE PLC 2..31 // Disable all other tasks
M_SCtrl_ulSystemCommandCOS=$55AA55AA // Reset token for MASTER Unit
M_HCSC_HSF_RESET=1 // Reset bit, token required for reset to complete
M_CommErrorFlag=0
timer = 4000 msec // Reset Time-out Timer
WHILE (M_CommErrorFlag=0 AND M_HCSC_NSF_READY=0) // Wait for reset to complete
IF (timer<0) // Check for reset timeout
M_CommErrorFlag = 1
ENDIF
ENDWHILE

IF (M_CommErrorFlag=0) //
WHILE (M_CC0_RCX_COMM_COS_RUN=0) // wait for comm tasks to
// start on COMX modules
M_HCCC0_HCF_NETX_COS_ACK = M_HCCC0_HCF_NETX_COS_ACK ^ 1
// Toggle Communication Channel 0's Change of State Acknowledge bit in
// order to read the CC0_RCX_COMM_COS_RUN which is a part of Communication
// Channel 0 State Register
enable plc 2
ENDWHILE
ENDIF
DISABLE PLC 1
CLOSE

open plc2 clr
timer =4000 msec // Reset Time-out Timer
//M_CC0_RCX_APP_COS_APP_READY=1


IF (M_HCSC_HSF_HOST_COS_CMD    = M_HCSC_NSF_HOST_COS_ACK)
       M_CC0_RCX_APP_COS_BUS_ON=1 // Setting the Bus On flag for 1st ACC-72EX
       M_CC0_RCX_APP_COS_BUS_ON_ENABLE=1
  M_HCSC_HSF_HOST_COS_CMD      = M_HCSC_HSF_HOST_COS_CMD^1
ENDIF

timer = 1000 msec // Reset Time-out Timer


ENABLE PLC 28
ENABLE PLC 10
ENABLE PLC 11
ENABLE PLC 26
disable plc2
close

OPEN PLC 28 CLEAR
M_CC0_ulDeviceWatchdog = M_CC0_ulHostWatchdog // copies the host watchdog content
CLOSE


OPEN PLC 10 CLEAR
IF (M_HCCC0_HCF_PD0_OUT_CMD = M_HCCC0_NCF_PD0_OUT_ACK) // Making sure the ACK flag matches the
CMD
```

```
  M_HCCC0_HCF_PD0_OUT_CMD = M_HCCC0_HCF_PD0_OUT_CMD^1 // Toggling the CMD flag (^: XOR)
ENDIF

CLOSE

OPEN PLC 11 CLEAR
IF (M_HCCC0_NCF_PD0_IN_CMD = M_HCCC0_HCF_PD0_IN_ACK) // If CMD flag and ACK flags are
       M_HCCC0_HCF_PD0_IN_ACK = M_HCCC0_HCF_PD0_IN_ACK ^ 1 // toggle the acknowledge bit
EndIF

CLOSE



i5=2
Ena plc1

M90->y:$6C4C0,0,16           ;Write Address
m91->x:$6C4C0,0,16           ;Write Address
M92->y:$6CA60,0,16           ;Read Address
m93->x:$6CA60,0,16           ;Read Address
M94->Y:$00405A,0,20          ;address of M90
m95->Y:$00405B,0,20          ;address of m91
M96->Y:$00405C,0,20          ;address of M92
m97->Y:$00405D,0,20          ;address of m93
```

The above PLCs require the following header files:

```
// MacroNameDefinition_$6C000.h
#define M_SI_abCookie_0_             M5000
#define M_SI_abCookie_1_             M5001
#define M_SI_abCookie_2_             M5002
#define M_SI_abCookie_3_             M5003
#define M_SI_ulDpmTotalSize          M5004
#define M_SI_ulDeviceNumber          M5005
#define M_SI_ulSerialNumber          M5006
#define M_SI_ausHwOptions_0_         M5007
#define M_SI_ausHwOptions_1_         M5008
#define M_SI_ausHwOptions_2_         M5009
#define M_SI_ausHwOptions_3_         M5010
#define M_SI_usManufacturer          M5011
#define M_SI_usProductionDate        M5012
#define M_SI_ulLicenseFlags1         M5013
#define M_SI_ulLicenseFlags2         M5014
#define M_SI_usNetxLicenseID         M5015
#define M_SI_usNetxLicenseFlags              M5016
#define M_SI_usDeviceClass           M5017
#define M_SI_bHwRevision             M5018
#define M_SI_bHwCompatibility        M5019
#define M_SI_bDevIdNumber            M5020
#define M_SCI_bChannelType           M5021
#define M_SCI_bSizePositionOfHandshake               M5022
#define M_SCI_bNumberOfBlocks        M5023
#define M_SCI_ulSizeOfChannel        M5024
#define M_SCI_usSizeOfMailbox        M5025
#define M_SCI_usMailboxStartOffset           M5026
#define M_HCI_bChannelType           M5027
#define M_HCI_ulSizeOfChannel        M5028
#define M_CC0I_bChannelType          M5029
#define M_CC0I_bChannelId            M5030
#define M_CC0I_bSizePositionOfHandshake              M5031
#define M_CC0I_bNumberOfBlocks               M5032
#define M_CC0I_ulSizeOfChannel               M5033
#define M_CC0I_usCommunicationClass          M5034
#define M_CC0I_usProtocolClass               M5035
#define M_CC0I_usConformanceClass            M5036
#define M_CC1I_bChannelType          M5037
#define M_CC1I_bChannelId            M5038
#define M_CC1I_bSizePositionOfHandshake              M5039
```

```
#define M_CC1I_bNumberOfBlocks             M5040
#define M_CC1I_ulSizeOfChannel             M5041
#define M_CC1I_usCommunicationClass        M5042
#define M_CC1I_usProtocolClass             M5043
#define M_CC1I_usConformanceClass          M5044
#define M_CC2I_bChannelType          M5045
#define M_CC2I_bChannelId            M5046
#define M_CC2I_bSizePositionOfHandshake            M5047
#define M_CC2I_bNumberOfBlocks             M5048
#define M_CC2I_ulSizeOfChannel             M5049
#define M_CC2I_usCommunicationClass        M5050
#define M_CC2I_usProtocolClass             M5051
#define M_CC2I_usConformanceClass          M5052
#define M_CC3I_bChannelType          M5053
#define M_CC3I_bChannelId            M5054
#define M_CC3I_bSizePositionOfHandshake            M5055
#define M_CC3I_bNumberOfBlocks             M5056
#define M_CC3I_ulSizeOfChannel             M5057
#define M_CC3I_usCommunicationClass        M5058
#define M_CC3I_usProtocolClass             M5059
#define M_CC3I_usConformanceClass          M5060
#define M_AC0I_bChannelType          M5061
#define M_AC0I_bChannelId            M5062
#define M_AC0I_bSizePositionOfHandshake            M5063
#define M_AC0I_bNumberOfBlocks             M5064
#define M_AC0I_ulSizeOfChannel             M5065
#define M_AC1I_bChannelType          M5066
#define M_AC1I_bChannelId            M5067
#define M_AC1I_bSizePositionOfHandshake            M5068
#define M_AC1I_bNumberOfBlocks             M5069
#define M_AC1I_ulSizeOfChannel             M5070
#define M_SCtrl_ulSystemCommandCOS         M5071
#define M_SStat_ulSystemCOS          M5072
#define M_SStat_ulSystemStatus             M5073
#define M_SStat_ulSystemError        M5074
#define M_SStat_ulBootError          M5075
#define M_SStat_ulTimeSinceStart           M5076
#define M_SStat_usCpuLoad            M5077
#define M_SStat_ulHWFeatures         M5078
#define M_SSMB_usPackagesAccepted          M5079
#define M_SSMB_ulDest          M5080
#define M_SSMB_ulSrc           M5081
#define M_SSMB_ulDestId              M5082
#define M_SSMB_ulSrcId         M5083
#define M_SSMB_ulLen           M5084
#define M_SSMB_ulId            M5085
#define M_SSMB_ulState         M5086
#define M_SSMB_ulCmd           M5087
#define M_SSMB_ulExt           M5088
#define M_SSMB_ulRout          M5089
#define M_SSMB_ultData0              M5090
#define M_SSMB_ultData1              M5091
#define M_SSMB_ultData2              M5092
#define M_SSMB_ultData3              M5093
#define M_SSMB_ultData4              M5094
#define M_SSMB_ultData5              M5095
#define M_SSMB_ultData6              M5096
#define M_SSMB_ultData7              M5097
#define M_SSMB_ultData8              M5098
#define M_SSMB_ultData9              M5099
#define M_SSMB_ultData10             M5100
#define M_SSMB_ultData11             M5101
#define M_SSMB_ultData12             M5102
#define M_SSMB_ultData13             M5103
#define M_SSMB_ultData14             M5104
#define M_SSMB_ultData15             M5105
#define M_SSMB_ultData16             M5106
#define M_SSMB_ultData17             M5107
#define M_SSMB_ultData18             M5108
#define M_SSMB_ultData19             M5109
#define M_SSMB_ultData20             M5110
```

```
#define M_SRMB_usWaitingPackages            M5111
#define M_SRMB_ulDest           M5112
#define M_SRMB_ulSrc            M5113
#define M_SRMB_ulDestId                 M5114
#define M_SRMB_ulSrcId          M5115
#define M_SRMB_ulLen            M5116
#define M_SRMB_ulId             M5117
#define M_SRMB_ulState          M5118
#define M_SRMB_ulCmd            M5119
#define M_SRMB_ulExt            M5120
#define M_SRMB_ulRout           M5121
#define M_SRMB_ultData0                 M5122
#define M_SRMB_ultData1                 M5123
#define M_SRMB_ultData2                 M5124
#define M_SRMB_ultData3                 M5125
#define M_SRMB_ultData4                 M5126
#define M_SRMB_ultData5                 M5127
#define M_SRMB_ultData6                 M5128
#define M_SRMB_ultData7                 M5129
#define M_SRMB_ultData8                 M5130
#define M_SRMB_ultData9                 M5131
#define M_SRMB_ultData10                M5132
#define M_SRMB_ultData11                M5133
#define M_SRMB_ultData12                M5134
#define M_SRMB_ultData13                M5135
#define M_SRMB_ultData14                M5136
#define M_SRMB_ultData15                M5137
#define M_SRMB_ultData16                M5138
#define M_SRMB_ultData17                M5139
#define M_SRMB_ultData18                M5140
#define M_SRMB_ultData19                M5141
#define M_SRMB_ultData20                M5142
#define M_HCSC_bNetxFlags               M5143
#define M_HCSC_NSF_READY                M5144
#define M_HCSC_NSF_ERROR                M5145
#define M_HCSC_NSF_HOST_COS_ACK                 M5146
#define M_HCSC_NSF_NETX_COS_CMD                 M5147
#define M_HCSC_NSF_SEND_MBX_ACK                 M5148
#define M_HCSC_NSF_RECV_MBX_CMD                 M5149
#define M_HCSC_bHostFlags               M5150
#define M_HCSC_HSF_RESET                M5151
#define M_HCSC_HSF_BOOTSTART            M5152
#define M_HCSC_HSF_HOST_COS_CMD                 M5153
#define M_HCSC_HSF_NETX_COS_ACK                 M5154
#define M_HCSC_HSF_SEND_MBX_CMD                 M5155
#define M_HCSC_HSF_RECV_MBX_ACK                 M5156
#define M_HCCC0_usNetxFlags             M5157
#define M_HCCC0_NCF_COMMUNICATING               M5158
#define M_HCCC0_NCF_ERROR               M5159
#define M_HCCC0_NCF_HOST_COS_ACK                M5160
#define M_HCCC0_NCF_NETX_COS_CMD                M5161
#define M_HCCC0_NCF_SEND_MBX_ACK                M5162
#define M_HCCC0_NCF_RECV_MBX_CMD                M5163
#define M_HCCC0_NCF_PD0_OUT_ACK                 M5164
#define M_HCCC0_NCF_PD0_IN_CMD                  M5165
#define M_HCCC0_NCF_PD1_OUT_ACK                 M5166
#define M_HCCC0_NCF_PD1_IN_CMD                  M5167
#define M_HCCC0_usHostFlags             M5168
#define M_HCCC0_HCF_HOST_COS_CMD                M5169
#define M_HCCC0_HCF_NETX_COS_ACK                M5170
#define M_HCCC0_HCF_SEND_MBX_CMD                M5171
#define M_HCCC0_HCF_RECV_MBX_ACK                M5172
#define M_HCCC0_HCF_PD0_OUT_CMD                 M5173
#define M_HCCC0_HCF_PD0_IN_ACK                  M5174
#define M_HCCC0_HCF_PD1_OUT_CMD                 M5175
#define M_HCCC0_HCF_PD1_IN_ACK                  M5176
#define M_HCCC1_usNetxFlags             M5177
#define M_HCCC1_NCF_COMMUNICATING               M5178
#define M_HCCC1_NCF_ERROR               M5179
#define M_HCCC1_NCF_HOST_COS_ACK                M5180
#define M_HCCC1_NCF_NETX_COS_CMD                M5181
```

```
#define M_HCCC1_NCF_SEND_MBX_ACK             M5182
#define M_HCCC1_NCF_RECV_MBX_CMD             M5183
#define M_HCCC1_NCF_PD0_OUT_ACK              M5184
#define M_HCCC1_NCF_PD0_IN_CMD               M5185
#define M_HCCC1_NCF_PD1_OUT_ACK              M5186
#define M_HCCC1_NCF_PD1_IN_CMD               M5187
#define M_HCCC1_usHostFlags          M5188
#define M_HCCC1_HCF_HOST_COS_CMD             M5189
#define M_HCCC1_HCF_NETX_COS_ACK             M5190
#define M_HCCC1_HCF_SEND_MBX_CMD             M5191
#define M_HCCC1_HCF_RECV_MBX_ACK             M5192
#define M_HCCC1_HCF_PD0_OUT_CMD              M5193
#define M_HCCC1_HCF_PD0_IN_ACK               M5194
#define M_HCCC1_HCF_PD1_OUT_CMD              M5195
#define M_HCCC1_HCF_PD1_IN_ACK               M5196
#define M_HCCC2_usNetxFlags          M5197
#define M_HCCC2_NCF_COMMUNICATING            M5198
#define M_HCCC2_NCF_ERROR            M5199
#define M_HCCC2_NCF_HOST_COS_ACK             M5200
#define M_HCCC2_NCF_NETX_COS_CMD             M5201
#define M_HCCC2_NCF_SEND_MBX_ACK             M5202
#define M_HCCC2_NCF_RECV_MBX_CMD             M5203
#define M_HCCC2_NCF_PD0_OUT_ACK              M5204
#define M_HCCC2_NCF_PD0_IN_CMD               M5205
#define M_HCCC2_NCF_PD1_OUT_ACK              M5206
#define M_HCCC2_NCF_PD1_IN_CMD               M5207
#define M_HCCC2_usHostFlags          M5208
#define M_HCCC2_HCF_HOST_COS_CMD             M5209
#define M_HCCC2_HCF_NETX_COS_ACK             M5210
#define M_HCCC2_HCF_SEND_MBX_CMD             M5211
#define M_HCCC2_HCF_RECV_MBX_ACK             M5212
#define M_HCCC2_HCF_PD0_OUT_CMD              M5213
#define M_HCCC2_HCF_PD0_IN_ACK               M5214
#define M_HCCC2_HCF_PD1_OUT_CMD              M5215
#define M_HCCC2_HCF_PD1_IN_ACK               M5216
#define M_HCCC3_usNetxFlags          M5217
#define M_HCCC3_NCF_COMMUNICATING            M5218
#define M_HCCC3_NCF_ERROR            M5219
#define M_HCCC3_NCF_HOST_COS_ACK             M5220
#define M_HCCC3_NCF_NETX_COS_CMD             M5221
#define M_HCCC3_NCF_SEND_MBX_ACK             M5222
#define M_HCCC3_NCF_RECV_MBX_CMD             M5223
#define M_HCCC3_NCF_PD0_OUT_ACK              M5224
#define M_HCCC3_NCF_PD0_IN_CMD               M5225
#define M_HCCC3_NCF_PD1_OUT_ACK              M5226
#define M_HCCC3_NCF_PD1_IN_CMD               M5227
#define M_HCCC3_usHostFlags          M5228
#define M_HCCC3_HCF_HOST_COS_CMD             M5229
#define M_HCCC3_HCF_NETX_COS_ACK             M5230
#define M_HCCC3_HCF_SEND_MBX_CMD             M5231
#define M_HCCC3_HCF_RECV_MBX_ACK             M5232
#define M_HCCC3_HCF_PD0_OUT_CMD              M5233
#define M_HCCC3_HCF_PD0_IN_ACK               M5234
#define M_HCCC3_HCF_PD1_OUT_CMD              M5235
#define M_HCCC3_HCF_PD1_IN_ACK               M5236
#define M_HCAC0_usNetxFlags          M5237
#define M_HCAC0_NCF_COMMUNICATING            M5238
#define M_HCAC0_NCF_ERROR            M5239
#define M_HCAC0_NCF_HOST_COS_ACK             M5240
#define M_HCAC0_NCF_NETX_COS_CMD             M5241
#define M_HCAC0_NCF_SEND_MBX_ACK             M5242
#define M_HCAC0_NCF_RECV_MBX_CMD             M5243
#define M_HCAC0_NCF_PD0_OUT_ACK              M5244
#define M_HCAC0_NCF_PD0_IN_CMD               M5245
#define M_HCAC0_NCF_PD1_OUT_ACK              M5246
#define M_HCAC0_NCF_PD1_IN_CMD               M5247
#define M_HCAC0_usHostFlags          M5248
#define M_HCAC0_HCF_HOST_COS_CMD             M5249
#define M_HCAC0_HCF_NETX_COS_ACK             M5250
#define M_HCAC0_HCF_SEND_MBX_CMD             M5251
#define M_HCAC0_HCF_RECV_MBX_ACK             M5252
```

```
#define M_HCAC0_HCF_PD0_OUT_CMD              M5253
#define M_HCAC0_HCF_PD0_IN_ACK               M5254
#define M_HCAC0_HCF_PD1_OUT_CMD              M5255
#define M_HCAC0_HCF_PD1_IN_ACK               M5256
#define M_HCAC1_usNetxFlags          M5257
#define M_HCAC1_NCF_COMMUNICATING            M5258
#define M_HCAC1_NCF_ERROR            M5259
#define M_HCAC1_NCF_HOST_COS_ACK             M5260
#define M_HCAC1_NCF_NETX_COS_CMD             M5261
#define M_HCAC1_NCF_SEND_MBX_ACK             M5262
#define M_HCAC1_NCF_RECV_MBX_CMD             M5263
#define M_HCAC1_NCF_PD0_OUT_ACK              M5264
#define M_HCAC1_NCF_PD0_IN_CMD               M5265
#define M_HCAC1_NCF_PD1_OUT_ACK              M5266
#define M_HCAC1_NCF_PD1_IN_CMD               M5267
#define M_HCAC1_usHostFlags          M5268
#define M_HCAC1_HCF_HOST_COS_CMD             M5269
#define M_HCAC1_HCF_NETX_COS_ACK             M5270
#define M_HCAC1_HCF_SEND_MBX_CMD             M5271
#define M_HCAC1_HCF_RECV_MBX_ACK             M5272
#define M_HCAC1_HCF_PD0_OUT_CMD              M5273
#define M_HCAC1_HCF_PD0_IN_ACK               M5274
#define M_HCAC1_HCF_PD1_OUT_CMD              M5275
#define M_HCAC1_HCF_PD1_IN_ACK               M5276
#define M_CC0_RCX_APP_COS_APP_READY          M5277
#define M_CC0_RCX_APP_COS_BUS_ON             M5278
#define M_CC0_RCX_APP_COS_BUS_ON_ENABLE              M5279
#define M_CC0_RCX_APP_COS_INIT               M5280
#define M_CC0_RCX_APP_COS_INIT_ENABLE        M5281
#define M_CC0_RCX_APP_COS_LOCK_CFG           M5282
#define M_CC0_RCX_APP_COS_LOCK_CFG_ENA               M5283
#define M_CC0_RCX_APP_COS_DMA        M5284
#define M_CC0_RCX_APP_COS_DMA_ENABLE         M5285
#define M_CC0_ulDeviceWatchdog               M5286
#define M_CC0_RCX_COMM_COS_READY             M5287
#define M_CC0_RCX_COMM_COS_RUN               M5288
#define M_CC0_RCX_COMM_COS_BUS_ON            M5289
#define M_CC0_RCX_COMM_COS_CONFIG_LOCKED             M5290
#define M_CC0_RCX_COMM_COS_CONFIG_NEW        M5291
#define M_CC0_RCX_COMM_COS_RESTART_REQ               M5292
#define M_CC0_RCX_COMM_CO_REQ_ENA            M5293
#define M_CC0_RCX_COMM_COS_DMA               M5294
#define M_CC0_ulCommunicationState           M5295
#define M_CC0_ulCommunicationError           M5296
#define M_CC0_usVersion              M5297
#define M_CC0_usWatchdogTime         M5298
#define M_CC0_bPDInHskMode           M5299
#define M_CC0_bPDInSource            M5300
#define M_CC0_bPDOutHskMode          M5301
#define M_CC0_bPDOutSource           M5302
#define M_CC0_ulHostWatchdog         M5303
#define M_CC0_ulErrorCount           M5304
#define M_CC0_bErrorLogInd           M5305
#define M_CC0_bErrorPDInCnt          M5306
#define M_CC0_bErrorPDOutCnt         M5307
#define M_CC0_bErrorSyncCnt          M5308
#define M_CC0_bSyncHskMode           M5309
#define M_CC0_bSyncSource            M5310
#define M_CC0_ulSlaveState           M5311
#define M_CC0_ulSlaveErrLogInd               M5312
#define M_CC0_ulNumOfConfigSlaves            M5313
#define M_CC0_ulNumOfActiveSlaves            M5314
#define M_CC0_ulNumOfDiagSlaves              M5315
```

Next file:

```
// M-VariableDefinition_$6C000.pmc
CLOSE
END GAT
DEL GAT
#Include "MacroNameDefinition_$6C000.h"

M_SI_abCookie_0_->Y:$6C000,0,8
M_SI_abCookie_1_->Y:$6C000,8,8
M_SI_abCookie_2_->X:$6C000,0,8
M_SI_abCookie_3_->X:$6C000,8,8
M_SI_ulDpmTotalSize->DP:$6C001
M_SI_ulDeviceNumber->DP:$6C002
M_SI_ulSerialNumber->DP:$6C003
M_SI_ausHwOptions_0_->Y:$6C004,0,16
M_SI_ausHwOptions_1_->X:$6C004,0,16
M_SI_ausHwOptions_2_->Y:$6C005,0,16
M_SI_ausHwOptions_3_->X:$6C005,0,16
M_SI_usManufacturer->Y:$6C006,0,16
M_SI_usProductionDate->X:$6C006,0,16
M_SI_ulLicenseFlags1->DP:$6C007
M_SI_ulLicenseFlags2->DP:$6C008
M_SI_usNetxLicenseID->Y:$6C009,0,16
M_SI_usNetxLicenseFlags->X:$6C009,0,16
M_SI_usDeviceClass->Y:$6C00A,0,16
M_SI_bHwRevision->X:$6C00A,0,8
M_SI_bHwCompatibility->X:$6C00A,8,8
M_SI_bDevIdNumber->Y:$6C00B,0,8
M_SCI_bChannelType->Y:$6C00C,0,8
M_SCI_bSizePositionOfHandshake->X:$6C00C,0,8
M_SCI_bNumberOfBlocks->X:$6C00C,8,8
M_SCI_ulSizeOfChannel->DP:$6C00D
M_SCI_usSizeOfMailbox->Y:$6C00E,0,16
M_SCI_usMailboxStartOffset->X:$6C00E,0,16
M_HCI_bChannelType->Y:$6C010,0,8
M_HCI_ulSizeOfChannel->DP:$6C011
M_CC0I_bChannelType->Y:$6C014,0,8
M_CC0I_bChannelId->Y:$6C014,8,8
M_CC0I_bSizePositionOfHandshake->X:$6C014,0,8
M_CC0I_bNumberOfBlocks->X:$6C014,8,8
M_CC0I_ulSizeOfChannel->DP:$6C015
M_CC0I_usCommunicationClass->Y:$6C016,0,16
M_CC0I_usProtocolClass->X:$6C016,0,16
M_CC0I_usConformanceClass->Y:$6C017,0,16
M_CC1I_bChannelType->Y:$6C018,0,8
M_CC1I_bChannelId->Y:$6C018,8,8
M_CC1I_bSizePositionOfHandshake->X:$6C018,0,8
M_CC1I_bNumberOfBlocks->X:$6C018,8,8
M_CC1I_ulSizeOfChannel->DP:$6C019
M_CC1I_usCommunicationClass->Y:$6C01A,0,16
M_CC1I_usProtocolClass->X:$6C01A,0,16
M_CC1I_usConformanceClass->Y:$6C01B,0,16
M_CC2I_bChannelType->Y:$6C01C,0,8
M_CC2I_bChannelId->Y:$6C01C,8,8
M_CC2I_bSizePositionOfHandshake->X:$6C01C,0,8
M_CC2I_bNumberOfBlocks->X:$6C01C,8,8
M_CC2I_ulSizeOfChannel->DP:$6C01D
M_CC2I_usCommunicationClass->Y:$6C01E,0,16
M_CC2I_usProtocolClass->X:$6C01E,0,16
M_CC2I_usConformanceClass->Y:$6C01F,0,16
M_CC3I_bChannelType->Y:$6C020,0,8
M_CC3I_bChannelId->Y:$6C020,8,8
M_CC3I_bSizePositionOfHandshake->X:$6C020,0,8
M_CC3I_bNumberOfBlocks->X:$6C020,8,8
M_CC3I_ulSizeOfChannel->DP:$6C021
M_CC3I_usCommunicationClass->Y:$6C022,0,16
M_CC3I_usProtocolClass->X:$6C022,0,16
M_CC3I_usConformanceClass->Y:$6C023,0,16
M_AC0I_bChannelType->Y:$6C024,0,8
M_AC0I_bChannelId->Y:$6C024,8,8
```

```
M_AC0I_bSizePositionOfHandshake->X:$6C024,0,8
M_AC0I_bNumberOfBlocks->X:$6C024,8,8
M_AC0I_ulSizeOfChannel->DP:$6C025
M_AC1I_bChannelType->Y:$6C028,0,8
M_AC1I_bChannelId->Y:$6C028,8,8
M_AC1I_bSizePositionOfHandshake->X:$6C028,0,8
M_AC1I_bNumberOfBlocks->X:$6C028,8,8
M_AC1I_ulSizeOfChannel->DP:$6C029
M_SCtrl_ulSystemCommandCOS->DP:$6C02E
M_SStat_ulSystemCOS->DP:$6C030
M_SStat_ulSystemStatus->DP:$6C031
M_SStat_ulSystemError->DP:$6C032
M_SStat_ulBootError->DP:$6C033
M_SStat_ulTimeSinceStart->DP:$6C034
M_SStat_usCpuLoad->Y:$6C035,0,16
M_SStat_ulHWFeatures->DP:$6C036
M_SSMB_usPackagesAccepted->Y:$6C040,0,16
M_SSMB_ulDest->DP:$6C041
M_SSMB_ulSrc->DP:$6C042
M_SSMB_ulDestId->DP:$6C043
M_SSMB_ulSrcId->DP:$6C044
M_SSMB_ulLen->DP:$6C045
M_SSMB_ulId->DP:$6C046
M_SSMB_ulState->DP:$6C047
M_SSMB_ulCmd->DP:$6C048
M_SSMB_ulExt->DP:$6C049
M_SSMB_ulRout->DP:$6C04A
M_SSMB_ultData0->DP:$6C04B
M_SSMB_ultData1->DP:$6C04C
M_SSMB_ultData2->DP:$6C04D
M_SSMB_ultData3->DP:$6C04E
M_SSMB_ultData4->DP:$6C04F
M_SSMB_ultData5->DP:$6C050
M_SSMB_ultData6->DP:$6C051
M_SSMB_ultData7->DP:$6C052
M_SSMB_ultData8->DP:$6C053
M_SSMB_ultData9->DP:$6C054
M_SSMB_ultData10->DP:$6C055
M_SSMB_ultData11->DP:$6C056
M_SSMB_ultData12->DP:$6C057
M_SSMB_ultData13->DP:$6C058
M_SSMB_ultData14->DP:$6C059
M_SSMB_ultData15->DP:$6C05A
M_SSMB_ultData16->DP:$6C05B
M_SSMB_ultData17->DP:$6C05C
M_SSMB_ultData18->DP:$6C05D
M_SSMB_ultData19->DP:$6C05E
M_SSMB_ultData20->DP:$6C05F
M_SRMB_usWaitingPackages->Y:$6C060,0,16
M_SRMB_ulDest->DP:$6C061
M_SRMB_ulSrc->DP:$6C062
M_SRMB_ulDestId->DP:$6C063
M_SRMB_ulSrcId->DP:$6C064
M_SRMB_ulLen->DP:$6C065
M_SRMB_ulId->DP:$6C066
M_SRMB_ulState->DP:$6C067
M_SRMB_ulCmd->DP:$6C068
M_SRMB_ulExt->DP:$6C069
M_SRMB_ulRout->DP:$6C06A
M_SRMB_ultData0->DP:$6C06B
M_SRMB_ultData1->DP:$6C06C
M_SRMB_ultData2->DP:$6C06D
M_SRMB_ultData3->DP:$6C06E
M_SRMB_ultData4->DP:$6C06F
M_SRMB_ultData5->DP:$6C070
M_SRMB_ultData6->DP:$6C071
M_SRMB_ultData7->DP:$6C072
M_SRMB_ultData8->DP:$6C073
M_SRMB_ultData9->DP:$6C074
M_SRMB_ultData10->DP:$6C075
M_SRMB_ultData11->DP:$6C076
```

```
M_SRMB_ultData12->DP:$6C077
M_SRMB_ultData13->DP:$6C078
M_SRMB_ultData14->DP:$6C079
M_SRMB_ultData15->DP:$6C07A
M_SRMB_ultData16->DP:$6C07B
M_SRMB_ultData17->DP:$6C07C
M_SRMB_ultData18->DP:$6C07D
M_SRMB_ultData19->DP:$6C07E
M_SRMB_ultData20->DP:$6C07F
M_HCSC_bNetxFlags->X:$6C080,0,8
M_HCSC_NSF_READY->X:$6C080,0,1
M_HCSC_NSF_ERROR->X:$6C080,1,1
M_HCSC_NSF_HOST_COS_ACK->X:$6C080,2,1
M_HCSC_NSF_NETX_COS_CMD->X:$6C080,3,1
M_HCSC_NSF_SEND_MBX_ACK->X:$6C080,4,1
M_HCSC_NSF_RECV_MBX_CMD->X:$6C080,5,1
M_HCSC_bHostFlags->X:$6C080,8,8
M_HCSC_HSF_RESET->X:$6C080,8,1
M_HCSC_HSF_BOOTSTART->X:$6C080,9,1
M_HCSC_HSF_HOST_COS_CMD->X:$6C080,10,1
M_HCSC_HSF_NETX_COS_ACK->X:$6C080,11,1
M_HCSC_HSF_SEND_MBX_CMD->X:$6C080,12,1
M_HCSC_HSF_RECV_MBX_ACK->X:$6C080,13,1
M_HCCC0_usNetxFlags->Y:$6C082,0,16
M_HCCC0_NCF_COMMUNICATING->Y:$6C082,0,1
M_HCCC0_NCF_ERROR->Y:$6C082,1,1
M_HCCC0_NCF_HOST_COS_ACK->Y:$6C082,2,1
M_HCCC0_NCF_NETX_COS_CMD->Y:$6C082,3,1
M_HCCC0_NCF_SEND_MBX_ACK->Y:$6C082,4,1
M_HCCC0_NCF_RECV_MBX_CMD->Y:$6C082,5,1
M_HCCC0_NCF_PD0_OUT_ACK->Y:$6C082,6,1
M_HCCC0_NCF_PD0_IN_CMD->Y:$6C082,7,1
M_HCCC0_NCF_PD1_OUT_ACK->Y:$6C082,8,1
M_HCCC0_NCF_PD1_IN_CMD->Y:$6C082,9,1
M_HCCC0_usHostFlags->X:$6C082,0,16
M_HCCC0_HCF_HOST_COS_CMD->X:$6C082,2,1
M_HCCC0_HCF_NETX_COS_ACK->X:$6C082,3,1
M_HCCC0_HCF_SEND_MBX_CMD->X:$6C082,4,1
M_HCCC0_HCF_RECV_MBX_ACK->X:$6C082,5,1
M_HCCC0_HCF_PD0_OUT_CMD->X:$6C082,6,1
M_HCCC0_HCF_PD0_IN_ACK->X:$6C082,7,1
M_HCCC0_HCF_PD1_OUT_CMD->X:$6C082,8,1
M_HCCC0_HCF_PD1_IN_ACK->X:$6C082,9,1
M_HCCC1_usNetxFlags->Y:$6C083,0,16
M_HCCC1_NCF_COMMUNICATING->Y:$6C083,0,1
M_HCCC1_NCF_ERROR->Y:$6C083,1,1
M_HCCC1_NCF_HOST_COS_ACK->Y:$6C083,2,1
M_HCCC1_NCF_NETX_COS_CMD->Y:$6C083,3,1
M_HCCC1_NCF_SEND_MBX_ACK->Y:$6C083,4,1
M_HCCC1_NCF_RECV_MBX_CMD->Y:$6C083,5,1
M_HCCC1_NCF_PD0_OUT_ACK->Y:$6C083,6,1
M_HCCC1_NCF_PD0_IN_CMD->Y:$6C083,7,1
M_HCCC1_NCF_PD1_OUT_ACK->Y:$6C083,8,1
M_HCCC1_NCF_PD1_IN_CMD->Y:$6C083,9,1
M_HCCC1_usHostFlags->X:$6C083,0,16
M_HCCC1_HCF_HOST_COS_CMD->X:$6C083,2,1
M_HCCC1_HCF_NETX_COS_ACK->X:$6C083,3,1
M_HCCC1_HCF_SEND_MBX_CMD->X:$6C083,4,1
M_HCCC1_HCF_RECV_MBX_ACK->X:$6C083,5,1
M_HCCC1_HCF_PD0_OUT_CMD->X:$6C083,6,1
M_HCCC1_HCF_PD0_IN_ACK->X:$6C083,7,1
M_HCCC1_HCF_PD1_OUT_CMD->X:$6C083,8,1
M_HCCC1_HCF_PD1_IN_ACK->X:$6C083,9,1
M_HCCC2_usNetxFlags->Y:$6C084,0,16
M_HCCC2_NCF_COMMUNICATING->Y:$6C084,0,1
M_HCCC2_NCF_ERROR->Y:$6C084,1,1
M_HCCC2_NCF_HOST_COS_ACK->Y:$6C084,2,1
M_HCCC2_NCF_NETX_COS_CMD->Y:$6C084,3,1
M_HCCC2_NCF_SEND_MBX_ACK->Y:$6C084,4,1
M_HCCC2_NCF_RECV_MBX_CMD->Y:$6C084,5,1
M_HCCC2_NCF_PD0_OUT_ACK->Y:$6C084,6,1
```

```
M_HCCC2_NCF_PD0_IN_CMD->Y:$6C084,7,1
M_HCCC2_NCF_PD1_OUT_ACK->Y:$6C084,8,1
M_HCCC2_NCF_PD1_IN_CMD->Y:$6C084,9,1
M_HCCC2_usHostFlags->X:$6C084,0,16
M_HCCC2_HCF_HOST_COS_CMD->X:$6C084,2,1
M_HCCC2_HCF_NETX_COS_ACK->X:$6C084,3,1
M_HCCC2_HCF_SEND_MBX_CMD->X:$6C084,4,1
M_HCCC2_HCF_RECV_MBX_ACK->X:$6C084,5,1
M_HCCC2_HCF_PD0_OUT_CMD->X:$6C084,6,1
M_HCCC2_HCF_PD0_IN_ACK->X:$6C084,7,1
M_HCCC2_HCF_PD1_OUT_CMD->X:$6C084,8,1
M_HCCC2_HCF_PD1_IN_ACK->X:$6C084,9,1
M_HCCC3_usNetxFlags->Y:$6C085,0,16
M_HCCC3_NCF_COMMUNICATING->Y:$6C085,0,1
M_HCCC3_NCF_ERROR->Y:$6C085,1,1
M_HCCC3_NCF_HOST_COS_ACK->Y:$6C085,2,1
M_HCCC3_NCF_NETX_COS_CMD->Y:$6C085,3,1
M_HCCC3_NCF_SEND_MBX_ACK->Y:$6C085,4,1
M_HCCC3_NCF_RECV_MBX_CMD->Y:$6C085,5,1
M_HCCC3_NCF_PD0_OUT_ACK->Y:$6C085,6,1
M_HCCC3_NCF_PD0_IN_CMD->Y:$6C085,7,1
M_HCCC3_NCF_PD1_OUT_ACK->Y:$6C085,8,1
M_HCCC3_NCF_PD1_IN_CMD->Y:$6C085,9,1
M_HCCC3_usHostFlags->X:$6C085,0,16
M_HCCC3_HCF_HOST_COS_CMD->X:$6C085,2,1
M_HCCC3_HCF_NETX_COS_ACK->X:$6C085,3,1
M_HCCC3_HCF_SEND_MBX_CMD->X:$6C085,4,1
M_HCCC3_HCF_RECV_MBX_ACK->X:$6C085,5,1
M_HCCC3_HCF_PD0_OUT_CMD->X:$6C085,6,1
M_HCCC3_HCF_PD0_IN_ACK->X:$6C085,7,1
M_HCCC3_HCF_PD1_OUT_CMD->X:$6C085,8,1
M_HCCC3_HCF_PD1_IN_ACK->X:$6C085,9,1
M_HCAC0_usNetxFlags->Y:$6C086,0,16
M_HCAC0_NCF_COMMUNICATING->Y:$6C086,0,1
M_HCAC0_NCF_ERROR->Y:$6C086,1,1
M_HCAC0_NCF_HOST_COS_ACK->Y:$6C086,2,1
M_HCAC0_NCF_NETX_COS_CMD->Y:$6C086,3,1
M_HCAC0_NCF_SEND_MBX_ACK->Y:$6C086,4,1
M_HCAC0_NCF_RECV_MBX_CMD->Y:$6C086,5,1
M_HCAC0_NCF_PD0_OUT_ACK->Y:$6C086,6,1
M_HCAC0_NCF_PD0_IN_CMD->Y:$6C086,7,1
M_HCAC0_NCF_PD1_OUT_ACK->Y:$6C086,8,1
M_HCAC0_NCF_PD1_IN_CMD->Y:$6C086,9,1
M_HCAC0_usHostFlags->X:$6C086,0,16
M_HCAC0_HCF_HOST_COS_CMD->X:$6C086,2,1
M_HCAC0_HCF_NETX_COS_ACK->X:$6C086,3,1
M_HCAC0_HCF_SEND_MBX_CMD->X:$6C086,4,1
M_HCAC0_HCF_RECV_MBX_ACK->X:$6C086,5,1
M_HCAC0_HCF_PD0_OUT_CMD->X:$6C086,6,1
M_HCAC0_HCF_PD0_IN_ACK->X:$6C086,7,1
M_HCAC0_HCF_PD1_OUT_CMD->X:$6C086,8,1
M_HCAC0_HCF_PD1_IN_ACK->X:$6C086,9,1
M_HCAC1_usNetxFlags->Y:$6C087,0,16
M_HCAC1_NCF_COMMUNICATING->Y:$6C087,0,1
M_HCAC1_NCF_ERROR->Y:$6C087,1,1
M_HCAC1_NCF_HOST_COS_ACK->Y:$6C087,2,1
M_HCAC1_NCF_NETX_COS_CMD->Y:$6C087,3,1
M_HCAC1_NCF_SEND_MBX_ACK->Y:$6C087,4,1
M_HCAC1_NCF_RECV_MBX_CMD->Y:$6C087,5,1
M_HCAC1_NCF_PD0_OUT_ACK->Y:$6C087,6,1
M_HCAC1_NCF_PD0_IN_CMD->Y:$6C087,7,1
M_HCAC1_NCF_PD1_OUT_ACK->Y:$6C087,8,1
M_HCAC1_NCF_PD1_IN_CMD->Y:$6C087,9,1
M_HCAC1_usHostFlags->X:$6C087,0,16
M_HCAC1_HCF_HOST_COS_CMD->X:$6C087,2,1
M_HCAC1_HCF_NETX_COS_ACK->X:$6C087,3,1
M_HCAC1_HCF_SEND_MBX_CMD->X:$6C087,4,1
M_HCAC1_HCF_RECV_MBX_ACK->X:$6C087,5,1
M_HCAC1_HCF_PD0_OUT_CMD->X:$6C087,6,1
M_HCAC1_HCF_PD0_IN_ACK->X:$6C087,7,1
M_HCAC1_HCF_PD1_OUT_CMD->X:$6C087,8,1
```

```
M_HCAC1_HCF_PD1_IN_ACK->X:$6C087,9,1
M_CC0_RCX_APP_COS_APP_READY->Y:$6C0C2,0,1
M_CC0_RCX_APP_COS_BUS_ON->Y:$6C0C2,1,1
M_CC0_RCX_APP_COS_BUS_ON_ENABLE->Y:$6C0C2,2,1
M_CC0_RCX_APP_COS_INIT->Y:$6C0C2,3,1
M_CC0_RCX_APP_COS_INIT_ENABLE->Y:$6C0C2,4,1
M_CC0_RCX_APP_COS_LOCK_CFG->Y:$6C0C2,5,1
M_CC0_RCX_APP_COS_LOCK_CFG_ENA->Y:$6C0C2,6,1
M_CC0_RCX_APP_COS_DMA->Y:$6C0C2,7,1
M_CC0_RCX_APP_COS_DMA_ENABLE->Y:$6C0C2,8,1
M_CC0_ulDeviceWatchdog->DP:$6C0C3
M_CC0_RCX_COMM_COS_READY->Y:$6C0C4,0,1
M_CC0_RCX_COMM_COS_RUN->Y:$6C0C4,1,1
M_CC0_RCX_COMM_COS_BUS_ON->Y:$6C0C4,2,1
M_CC0_RCX_COMM_COS_CONFIG_LOCKED->Y:$6C0C4,3,1
M_CC0_RCX_COMM_COS_CONFIG_NEW->Y:$6C0C4,4,1
M_CC0_RCX_COMM_COS_RESTART_REQ->Y:$6C0C4,5,1
M_CC0_RCX_COMM_CO_REQ_ENA->Y:$6C0C4,6,1
M_CC0_RCX_COMM_COS_DMA->Y:$6C0C4,7,1
M_CC0_ulCommunicationState->DP:$6C0C5
M_CC0_ulCommunicationError->DP:$6C0C6
M_CC0_usVersion->Y:$6C0C7,0,16
M_CC0_usWatchdogTime->X:$6C0C7,0,16
M_CC0_bPDInHskMode->Y:$6C0C8,0,8
M_CC0_bPDInSource->Y:$6C0C8,8,8
M_CC0_bPDOutHskMode->X:$6C0C8,0,8
M_CC0_bPDOutSource->X:$6C0C8,8,8
M_CC0_ulHostWatchdog->DP:$6C0C9
M_CC0_ulErrorCount->DP:$6C0CA
M_CC0_bErrorLogInd->Y:$6C0CB,0,8
M_CC0_bErrorPDInCnt->Y:$6C0CB,8,8
M_CC0_bErrorPDOutCnt->X:$6C0CB,0,8
M_CC0_bErrorSyncCnt->X:$6C0CB,8,8
M_CC0_bSyncHskMode->Y:$6C0CC,0,8
M_CC0_bSyncSource->Y:$6C0CC,8,8
M_CC0_ulSlaveState->DP:$6C0CE
M_CC0_ulSlaveErrLogInd->DP:$6C0CF
M_CC0_ulNumOfConfigSlaves->DP:$6C0D0
M_CC0_ulNumOfActiveSlaves->DP:$6C0D1
M_CC0_ulNumOfDiagSlaves->DP:$6C0D2
```

APPENDIX B – TURBO PMAC MEMORY MAPS

	PROFIBUS-DP Master	PROFIBUS-DP Slave	DeviceNet Master	DeviceNet Slave	CANopen Master	CANopen Slave	CC-Link Slave	EtherCAT Master	EtherCAT Slave	EtherNet/IP Scanner/Master	EtherNet/IP Adapter/Slave	Open Modbus/TCP	PROFINET IO Controller/Master	PROFINET IO Device/Slave
ACC-72EX Address	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000
netX Identification	netX	netX	netX	netX	netX	netX	netX	netX	netX	netX	netX	netX	netX	netX
Dual-Port Memory Size	16384 bytes	8192 bytes	16384 bytes	8192 bytes	65536 bytes	8192 bytes	8192 bytes	16384 bytes	16384 bytes	65536 bytes	16384 bytes	16384 bytes	32768 bytes	32768 bytes
Device Number	1532410	1562420	1532510	1562520	1532500	1562540	1562740	1532100	1532100	1532100	1532100	1532100	1532100	1532100
Hardware Assembly Options														
Port 0	NOT CONNECTED	PROFIBUS	NOT CONNECTED	DEVICENET	NOT CONNECTED	CAN	CC-LINK	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)
Port 1	NOT CONNECTED	NOT AVAILABLE	NOT CONNECTED	NOT AVAILABLE	NOT CONNECTED	NOT AVAILABLE	NOT AVAILABLE	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)
Port 2	PROFIBUS	NOT AVAILABLE	DEVICENET	NOT AVAILABLE	CAN	NOT AVAILABLE	NOT AVAILABLE	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED
Port 3	NOT CONNECTED	NOT AVAILABLE	NOT CONNECTED	NOT AVAILABLE	NOT CONNECTED	NOT AVAILABLE	NOT AVAILABLE	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED
Hilscher Module License Information	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) 1 Master License	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) 1 Master License	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) 1 Master License	Unlimited number of master licenses	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) Unlimited number of master licenses	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) 1 Master License	Unlimited number of master licenses	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) 1 Master License	Unlimited number of master licenses
Tool License Information	(SYCON.net)		(SYCON.net)		(SYCON.net)			(SYCON.net)		(SYCON.net)			(SYCON.net)	
Device Class	COMX 100	COMX 10	COMX 100	COMX 10	COMX 100	COMX 10	COMX 10	COMX 100	COMX 100	COMX 100	COMX 100	COMX 100	COMX 100	COMX 100
+ Block 0														
Channel Type	System	System	System	System	System	System	System	System	System	System	System	System	System	System
Size of Channel	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes
Channel Start Address	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000
Position of Handshake Cells	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL
netX System Flags Adress	X:$6C080,0,8	X:$6C080,0,8	X:$6C080,0,8	X:$6C080,0,8	X:$6C080,0,8	X:$6C080,0,8	X:$6C080,0,8	X:$6C080,0,8	X:$6C080,0,8	X:$6C080,0,8	X:$6C080,0,8	X:$6C080,0,8	X:$6C080,0,8	X:$6C080,0,8
Host System Flags Adress	X:$6C080,8,8	X:$6C080,8,8	X:$6C080,8,8	X:$6C080,8,8	X:$6C080,8,8	X:$6C080,8,8	X:$6C080,8,8	X:$6C080,8,8	X:$6C080,8,8	X:$6C080,8,8	X:$6C080,8,8	X:$6C080,8,8	X:$6C080,8,8	X:$6C080,8,8
Size of Handshake Cells	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS
Size of Mailbox	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes
Mailbox Start address	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040
Number of Subblocks	5	5	5	5	5	5	5	5	5	5	5	5	5	5
--- Subblock 0	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS
Size	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes
Start Offset	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000	$6C000
Transfer Direction	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
--- Subblock 1	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL
Size	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes
Start Offset	$6C02E	$6C02E	$6C02E	$6C02E	$6C02E	$6C02E	$6C02E	$6C02E	$6C02E	$6C02E	$6C02E	$6C02E	$6C02E	$6C02E
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
--- Subblock 2	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS
Size	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes
Start Offset	$6C030	$6C030	$6C030	$6C030	$6C030	$6C030	$6C030	$6C030	$6C030	$6C030	$6C030	$6C030	$6C030	$6C030
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
--- Subblock 3	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX
Size	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes
Start Offset	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040	$6C040
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
Handshake Bit	4	4	4	4	4	4	4	4	4	4	4	4	4	4
--- Subblock 4	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX
Size	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes
Start Offset	$6C060	$6C060	$6C060	$6C060	$6C060	$6C060	$6C060	$6C060	$6C060	$6C060	$6C060	$6C060	$6C060	$6C060
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN
Handshake Bit	5	5	5	5	5	5	5	5	5	5	5	5	5	5
+ Block 1														
Channel Type	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake
Size of Channel	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes
Channel Start Address	$6C080	$6C080	$6C080	$6C080	$6C080	$6C080	$6C080	$6C080	$6C080	$6C080	$6C080	$6C080	$6C080	$6C080
+ Block 2														
Channel Type	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication
Size of Channel	15616 bytes	7424 bytes	15616 bytes	7424 bytes	15616 bytes	7424 bytes	7424 bytes	15616 bytes	15616 bytes	15616 bytes	15616 bytes	15616 bytes	15616 bytes	15616 bytes
Channel Start Address	$6C0C0	$6C0C0	$6C0C0	$6C0C0	$6C0C0	$6C0C0	$6C0C0	$6C0C0	$6C0C0	$6C0C0	$6C0C0	$6C0C0	$6C0C0	$6C0C0
Position of Handshake Cells	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL
Size of Handshake Cells	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS
NetX Handshake Register	Y:$6C082,0,16	Y:$6C082,0,16	Y:$6C082,0,16	Y:$6C082,0,16	Y:$6C082,0,16	Y:$6C082,0,16	Y:$6C082,0,16	Y:$6C082,0,16	Y:$6C082,0,16	Y:$6C082,0,16	Y:$6C082,0,16	Y:$6C082,0,16	Y:$6C082,0,16	Y:$6C082,0,16
Host Handshake Register	X:$6C082,0,16	X:$6C082,0,16	X:$6C082,0,16	X:$6C082,0,16	X:$6C082,0,16	X:$6C082,0,16	X:$6C082,0,16	X:$6C082,0,16	X:$6C082,0,16	X:$6C082,0,16	X:$6C082,0,16	X:$6C082,0,16	X:$6C082,0,16	X:$6C082,0,16
Communication Class	MASTER	SLAVE	MASTER	SLAVE	MASTER	SLAVE	SLAVE	MASTER	SLAVE	SCANNER	ADAPTER	MESSAGING	IO-CONTROLLER	IO-DEVICE
Protocol Class	Managing Node	Managing Node	Server	Server	Scanner	Scanner	Adapter	Io-Controller	Io-Controller	Io-Device	Io-Device	Combination Firmware	Programmable Logic Controller (Plc)	Programmable Logic Controller (Plc)
Conformance Class	0	0	0	0	0	0	0	0	0	0	0	0	0	67
Number of Subblocks	9	9	9	9	9	9	9	9	9	9	9	9	9	9
--- Subblock 0	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL
Size	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes
Start Offset	$6C0C2	$6C0C2	$6C0C2	$6C0C2	$6C0C2	$6C0C2	$6C0C2	$6C0C2	$6C0C2	$6C0C2	$6C0C2	$6C0C2	$6C0C2	$6C0C2
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
--- Subblock 1	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS
Size	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes
Start Offset	$6C0C4	$6C0C4	$6C0C4	$6C0C4	$6C0C4	$6C0C4	$6C0C4	$6C0C4	$6C0C4	$6C0C4	$6C0C4	$6C0C4	$6C0C4	$6C0C4
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
--- Subblock 2	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS
Size	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes
Start Offset	$6C0D4	$6C0D4	$6C0D4	$6C0D4	$6C0D4	$6C0D4	$6C0D4	$6C0D4	$6C0D4	$6C0D4	$6C0D4	$6C0D4	$6C0D4	$6C0D4
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
--- Subblock 3	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX
Size	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes
Start Offset	$6C140	$6C140	$6C140	$6C140	$6C140	$6C140	$6C140	$6C140	$6C140	$6C140	$6C140	$6C140	$6C140	$6C140
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
Handshake Bit	4	4	4	4	4	4	4	4	4	4	4	4	4	4

	PROFIBUS-DP Master	PROFIBUS-DP Slave	DeviceNet Master	DeviceNet Slave	CANopen Master	CANopen Slave	CC-Link Slave	EtherCAT Master	EtherCAT Slave	EtherNet/IP Scanner/Master	EtherNet/IP Adapter/Slave	Open Modbus/TCP	PROFINET IO Controller/Master	PROFINET IO Device/Slave
|--- Subblock 4	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX
Size	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes
Start Offset	$6C2D0	$6C2D0	$6C2D0	$6C2D0	$6C2D0	$6C2D0	$6C2D0	$6C2D0	$6C2D0	$6C2D0	$6C2D0	$6C2D0	$6C2D0	$6C2D0
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN
Handshake Bit	5	5	5	5	5	5	5	5	5	5	5	5	5	5
|--- Subblock 5	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE
Size	5760 bytes	1536 bytes	5760 bytes	1536 bytes	5760 bytes	1536 bytes	1536 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes
Start Offset	$6C4C0	$6C4C0	$6C4C0	$6C4C0	$6C4C0	$6C4C0	$6C4C0	$6C4C0	$6C4C0	$6C4C0	$6C4C0	$6C4C0	$6C4C0	$6C4C0
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
Handshake Bit	6	6	6	6	6	6	6	6	6	6	6	6	6	6
|--- Subblock 6	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE
Size	5760 bytes	1536 bytes	5760 bytes	1536 bytes	5760 bytes	1536 bytes	1536 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes
Start Offset	$6CA60	$6C640	$6CA60	$6C640	$6CA60	$6C640	$6C640	$6CA60	$6CA60	$6CA60	$6CA60	$6CA60	$6CA60	$6CA60
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
Handshake Bit	7	7	7	7	7	7	7	7	7	7	7	7	7	7
|--- Subblock 7	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE
Size	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes
Start Offset	$6C460	$6C460	$6C460	$6C460	$6C460	$6C460	$6C460	$6C460	$6C460	$6C460	$6C460	$6C460	$6C460	$6C460
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
Handshake Bit	8	8	8	8	8	8	8	8	8	8	8	8	8	8
|--- Subblock 8	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE
Size	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes
Start Offset	$6C470	$6C470	$6C470	$6C470	$6C470	$6C470	$6C470	$6C470	$6C470	$6C470	$6C470	$6C470	$6C470	$6C470
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
Handshake Bit	9	9	9	9	9	9	9	9	9	9	9	9	9	9
+ Block 3														
Channel Type	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Communication	Undefined	Undefined	Undefined	Communication
Size of Channel	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	15616 bytes	0 bytes	0 bytes	0 bytes	15616 bytes
Channel Start Address	$6D000	$6C800	$6D000	$6C800	$6D000	$6C800	$6C800	$6D000	$6D000	$6D000	$6D000	$6D000	$6D000	$6D000
Position of Handshake Cells	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	IN HANDSHAKE CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	IN HANDSHAKE CHANNEL
Size of Handshake Cells	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	16 BITS	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	16 BITS
NetX Handshake Register	X:$6D000	X:$6C800	X:$6D000	X:$6C800	X:$6D000	X:$6C800	X:$6C800	X:$6D000	X:$6D000	Y:$6C083,0,16	X:$6D000	X:$6D000	X:$6D000	Y:$6C083,0,16
Host Handshake Register	X:$6D000,8,0	X:$6C800,8,0	X:$6D000,8,0	X:$6C800,8,0	X:$6D000,8,0	X:$6C800,8,0	X:$6C800,8,0	X:$6D000,8,0	X:$6D000,8,0	X:$6C083,0,16	X:$6D000,8,0	X:$6D000,8,0	X:$6D000,8,0	X:$6C083,0,16
Communication Class	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	MESSAGING	UNDEFINED	UNDEFINED	UNDEFINED	MESSAGING
Protocol Class	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED
Conformance Class	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Number of Subblocks	0	0	0	0	0	0	0	0	0	9	0	0	0	9
|--- Subblock 0										CONTROL				CONTROL
Size										8 bytes				8 bytes
Start Offset										$6D002				$6D002
Transfer Direction										OUT (Host System to netX)				OUT (Host System to netX)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode										UNCONTROLLED				UNCONTROLLED
Handshake Bit										0				0
|--- Subblock 1										COMMON STATUS				COMMON STATUS
Size										64 bytes				64 bytes
Start Offset										$6D004				$6D004
Transfer Direction										IN (netX to Host System)				IN (netX to Host System)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode										UNCONTROLLED				UNCONTROLLED
Handshake Bit										0				0
|--- Subblock 2										EXTENDED STATUS				EXTENDED STATUS
Size										432 bytes				432 bytes
Start Offset										$6D014				$6D014
Transfer Direction										IN (netX to Host System)				IN (netX to Host System)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode										UNCONTROLLED				UNCONTROLLED
Handshake Bit										0				0
|--- Subblock 3										MAILBOX				MAILBOX
Size										1600 bytes				1600 bytes
Start Offset										$6D080				$6D080
Transfer Direction										OUT (Host System to netX)				OUT (Host System to netX)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode										BUFFERED, HOST CONTROLLED				BUFFERED, HOST CONTROLLED
Handshake Bit										4				4
|--- Subblock 4										MAILBOX				MAILBOX
Size										1600 bytes				1600 bytes
Start Offset										$6D210				$6D210
Transfer Direction										IN (netX to Host System)				IN (netX to Host System)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode										UNKNOWN				UNKNOWN
Handshake Bit										5				5
|--- Subblock 5										PROCESS DATA IMAGE				PROCESS DATA IMAGE
Size										5760 bytes				5760 bytes
Start Offset										$6D400				$6D400
Transfer Direction										OUT (Host System to netX)				OUT (Host System to netX)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode										BUFFERED, HOST CONTROLLED				BUFFERED, HOST CONTROLLED
Handshake Bit										6				6
|--- Subblock 6										PROCESS DATA IMAGE				PROCESS DATA IMAGE
Size										5760 bytes				5760 bytes
Start Offset										$6D9A0				$6D9A0
Transfer Direction										IN (netX to Host System)				IN (netX to Host System)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode										BUFFERED, HOST CONTROLLED				BUFFERED, HOST CONTROLLED
Handshake Bit										7				7
|--- Subblock 7										HIGH PRIORITY DATA IMAGE				HIGH PRIORITY DATA IMAGE
Size										64 bytes				64 bytes
Start Offset										$6D3A0				$6D3A0
Transfer Direction										OUT (Host System to netX)				OUT (Host System to netX)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode										BUFFERED, HOST CONTROLLED				BUFFERED, HOST CONTROLLED
Handshake Bit										8				8
|--- Subblock 8										HIGH PRIORITY DATA IMAGE				HIGH PRIORITY DATA IMAGE
Size										64 bytes				64 bytes
Start Offset										$6D3B0				$6D3B0
Transfer Direction										IN (netX to Host System)				IN (netX to Host System)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode										BUFFERED, HOST CONTROLLED				BUFFERED, HOST CONTROLLED
Handshake Bit										9				9
+ Block 4														
Channel Type	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
Size of Channel	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes

	PROFIBUS-DP Master	PROFIBUS-DP Slave	DeviceNet Master	DeviceNet Slave	CANopen Master	CANopen Slave	CC-Link Slave	EtherCAT Master	EtherCAT Slave	EtherNet/IP Scanner/Master	EtherNet/IP Adapter/Slave	Open Modbus/TCP	PROFINET IO Controller/Master	PROFINET IO Device/Slave
| Channel Start Address	$6D000	$6C800	$6D000	$6C800	$6D000	$6C800	$6C800	$6D000	$6D000	$6DF40	$6D000	$6D000	$6D000	$6DF40
| Position of Handshake Cells	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL
| Size of Handshake Cells	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE
| NetX Handshake Register	X:$6D000	X:$6C800	X:$6D000	X:$6C800	X:$6D000	X:$6C800	X:$6C800	X:$6D000	X:$6D000	X:$6DF40	X:$6D000	X:$6D000	X:$6D000	X:$6DF40
| Host Handshake Register	X:$6D000,8,0	X:$6C800,8,0	X:$6D000,8,0	X:$6C800,8,0	X:$6D000,8,0	X:$6C800,8,0	X:$6C800,8,0	X:$6D000,8,0	X:$6D000,8,0	X:$6DF40,8,0	X:$6D000,8,0	X:$6D000,8,0	X:$6D000,8,0	X:$6DF40,8,0
| Communication Class	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED
| Protocol Class	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED
| Conformance Class	0	0	0	0	0	0	0	0	0	0	0	0	0	0
| Number of Subblocks	0	0	0	0	0	0	0	0	0	0	0	0	0	0
|														
+ Block 5														
| Channel Type	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
| Size of Channel	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes
| Channel Start Address	$6D000	$6C800	$6D000	$6C800	$6D000	$6C800	$6C800	$6D000	$6D000	$6DF40	$6D000	$6D000	$6D000	$6DF40
| Position of Handshake Cells	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL
| Size of Handshake Cells	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE
| NetX Handshake Register	X:$6D000	X:$6C800	X:$6D000	X:$6C800	X:$6D000	X:$6C800	X:$6C800	X:$6D000	X:$6D000	X:$6DF40	X:$6D000	X:$6D000	X:$6D000	X:$6DF40
| Host Handshake Register	X:$6D000,8,0	X:$6C800,8,0	X:$6D000,8,0	X:$6C800,8,0	X:$6D000,8,0	X:$6C800,8,0	X:$6C800,8,0	X:$6D000,8,0	X:$6D000,8,0	X:$6DF40,8,0	X:$6D000,8,0	X:$6D000,8,0	X:$6D000,8,0	X:$6DF40,8,0
| Communication Class	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED
| Protocol Class	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED
| Conformance Class	0	0	0	0	0	0	0	0	0	0	0	0	0	0
| Number of Subblocks	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APPENDIX C – POWER PMAC MEMORY MAPS

	PROFIBUS-DP Master	PROFIBUS-DP Slave	DeviceNet Master	DeviceNet Slave	CANopen Master	CANopen Slave	CC-Link Slave	EtherCAT Master	EtherCAT Slave	EtherNet/IP Scanner/Master	EtherNet/IP Adapter/Slave	Open Modbus/TCP	PROFINET IO Controller/Master	PROFINET IO Device/Slave
ACC-72EX Address	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a
netX Identification	netX	netX	netX	netX	netX	netX	netX	netX	netX	netX	netX	netX	netX	netX
Dual-Port Memory Size	16384 bytes	8192 bytes	16384 bytes	8192 bytes	65536 bytes	8192 bytes	8192 bytes	16384 bytes	16384 bytes	65536 bytes	16384 bytes	16384 bytes	32768 bytes	32768 bytes
Device Number	1532410	1562420	1532510	1562520	1532500	1562540	1562740	1532100	1532100	1532100	1532100	1532100	1532100	1532100
Hardware Assembly Options														
Port 0	NOT CONNECTED	PROFIBUS	NOT CONNECTED	DEVICENET	NOT CONNECTED	CAN	CC-LINK	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)
Port 1	NOT CONNECTED	NOT AVAILABLE	NOT CONNECTED	NOT AVAILABLE	NOT CONNECTED	NOT AVAILABLE	NOT AVAILABLE	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)
Port 2	PROFIBUS	NOT AVAILABLE	DEVICENET	NOT AVAILABLE	CAN	NOT AVAILABLE	NOT AVAILABLE	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED
Port 3	NOT CONNECTED	NOT AVAILABLE	NOT CONNECTED	NOT AVAILABLE	NOT CONNECTED	NOT AVAILABLE	NOT AVAILABLE	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED
Hilscher Module License Information	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) 1 Master License	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) 1 Master License	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) 1 Master License	Unlimited number of master licenses	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) Unlimited number of master licenses	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) 1 Master License	Unlimited number of master licenses	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-Interface Master) (PROFINET IO RT Controller) (EtherCAT Master) (EtherNet/IP Scanner) (SERCOS III Master) 1 Master License	Unlimited number of master licenses
Tool License Information	(SYCON.net)		(SYCON.net)		(SYCON.net)			(SYCON.net)		(SYCON.net)			(SYCON.net)	
Device Class	COMX 100	COMX 10	COMX 100	COMX 10	COMX 100	COMX 10	COMX 10	COMX 100	COMX 100	COMX 100	COMX 100	COMX 100	COMX 100	COMX 100
+ Block 0														
Channel Type	System	System	System	System	System	System	System	System	System	System	System	System	System	System
Size of Channel	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes	512 bytes
Channel Start Address	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a
Position of Handshake Cells	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL
netX System Flags Address	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a	Acc72EX[i].Data8[514].a
Host System Flags Address	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a
Size of Handshake Cells	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS
Size of Mailbox	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes
Mailbox Start address	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a
Number of Subblocks	5	5	5	5	5	5	5	5	5	5	5	5	5	5
--- Subblock 0	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS
Size	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes
Start Offset	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a
Transfer Direction	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
--- Subblock 1	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL
Size	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes
Start Offset	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a	Acc72EX[i].Data8[184].a
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
--- Subblock 2	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS
Size	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes
Start Offset	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
--- Subblock 3	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX
Size	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes
Start Offset	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
Handshake Bit	4	4	4	4	4	4	4	4	4	4	4	4	4	4
--- Subblock 4	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX
Size	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes	128 bytes
Start Offset	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a	Acc72EX[i].Data8[384].a
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN
Handshake Bit	5	5	5	5	5	5	5	5	5	5	5	5	5	5
+ Block 1														
Channel Type	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake
Size of Channel	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes
Channel Start Address	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a	Acc72EX[i].Data8[512].a
+ Block 2														
Channel Type	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication	Communication
Size of Channel	15616 bytes	7424 bytes	15616 bytes	7424 bytes	15616 bytes	7424 bytes	7424 bytes	15616 bytes	15616 bytes	15616 bytes	15616 bytes	15616 bytes	15616 bytes	15616 bytes
Channel Start Address	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a
Position of Handshake Cells	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL
Size of Handshake Cells	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS	16 BITS
NetX Handshake Register	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a	Acc72EX[i].Data8[520].a
Host Handshake Register	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a	Acc72EX[i].Data8[522].a
Communication Class	MASTER	SLAVE	MASTER	SLAVE	MASTER	SLAVE	SLAVE	MASTER	SLAVE	SCANNER	ADAPTER	MESSAGING	IO-CONTROLLER	IO-DEVICE
Protocol Class	Managing Node	Managing Node	Server	Server	Scanner	Scanner	Adapter	Io-Controller	Io-Controller	Io-Device	Io-Device	Combination Firmware	Programmable Logic Controller (Plc)	Programmable Logic Controller (Plc)
Conformance Class	0	0	0	0	0	0	0	0	0	0	0	0	0	67
Number of Subblocks	9	9	9	9	9	9	9	9	9	9	9	9	9	9
--- Subblock 0	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL
Size	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes
Start Offset	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a	Acc72EX[i].Data8[768].a
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
--- Subblock 1	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS
Size	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes
Start Offset	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a	Acc72EX[i].Data8[784].a
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
--- Subblock 2	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS	EXTENDED STATUS
Size	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes	432 bytes
Start Offset	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a	Acc72EX[i].Data8[848].a
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
--- Subblock 3	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX
Size	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes
Start Offset	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a	Acc72EX[i].Data8[1280].a
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
Handshake Bit	4	4	4	4	4	4	4	4	4	4	4	4	4	4

	PROFIBUS-DP Master	PROFIBUS-DP Slave	DeviceNet Master	DeviceNet Slave	CANopen Master	CANopen Slave	CC-Link Slave	EtherCAT Master	EtherCAT Slave	EtherNet/IP Scanner/Master	EtherNet/IP Adapter/Slave	Open Modbus/TCP	PROFINET IO Controller/Master	PROFINET IO Device/Slave
|														
|--- **Subblock 4**	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX
| **Size**	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes	1600 bytes
| **Start Offset**	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a	Acc72EX[i].Data8[2280].a
| **Transfer Direction**	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
| **Transfer Type**	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
| **Handshake Mode**	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN
| **Handshake Bit**	5	5	5	5	5	5	5	5	5	5	5	5	5	5
|														
|--- **Subblock 5**	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE
| **Size**	5760 bytes	1536 bytes	5760 bytes	1536 bytes	5760 bytes	1536 bytes	1536 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes
| **Start Offset**	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a	Acc72EX[i].Data8[4480].a
| **Transfer Direction**	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
| **Transfer Type**	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
| **Handshake Mode**	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
| **Handshake Bit**	6	6	6	6	6	6	6	6	6	6	6	6	6	6
|														
|--- **Subblock 6**	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE
| **Size**	5760 bytes	1536 bytes	5760 bytes	1536 bytes	5760 bytes	1536 bytes	1536 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes
| **Start Offset**	Acc72EX[i].Data8[10240].a	Acc72EX[i].Data8[6016].a	Acc72EX[i].Data8[10240].a	Acc72EX[i].Data8[6016].a	Acc72EX[i].Data8[10240].a	Acc72EX[i].Data8[6016].a	Acc72EX[i].Data8[6016].a	Acc72EX[i].Data8[10240].a	Acc72EX[i].Data8[10240].a	Acc72EX[i].Data8[10240].a	Acc72EX[i].Data8[10240].a	Acc72EX[i].Data8[10240].a	Acc72EX[i].Data8[10240].a	Acc72EX[i].Data8[10240].a
| **Transfer Direction**	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
| **Transfer Type**	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
| **Handshake Mode**	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
| **Handshake Bit**	7	7	7	7	7	7	7	7	7	7	7	7	7	7
|														
|--- **Subblock 7**	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE
| **Size**	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes
| **Start Offset**	Acc72EX[i].Data8[16000].a	Acc72EX[i].Data8[7552].a	Acc72EX[i].Data8[16000].a	Acc72EX[i].Data8[7552].a	Acc72EX[i].Data8[16000].a	Acc72EX[i].Data8[7552].a	Acc72EX[i].Data8[7552].a	Acc72EX[i].Data8[16000].a	Acc72EX[i].Data8[16000].a	Acc72EX[i].Data8[16000].a	Acc72EX[i].Data8[16000].a	Acc72EX[i].Data8[16000].a	Acc72EX[i].Data8[16000].a	Acc72EX[i].Data8[16000].a
| **Transfer Direction**	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
| **Transfer Type**	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
| **Handshake Mode**	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
| **Handshake Bit**	8	8	8	8	8	8	8	8	8	8	8	8	8	8
|														
|--- **Subblock 8**	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE
Size	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes
Start Offset	Acc72EX[i].Data8[16064].a	Acc72EX[i].Data8[7616].a	Acc72EX[i].Data8[16064].a	Acc72EX[i].Data8[7616].a	Acc72EX[i].Data8[16064].a	Acc72EX[i].Data8[7616].a	Acc72EX[i].Data8[7616].a	Acc72EX[i].Data8[16064].a	Acc72EX[i].Data8[16064].a	Acc72EX[i].Data8[16064].a	Acc72EX[i].Data8[16064].a	Acc72EX[i].Data8[16064].a	Acc72EX[i].Data8[16064].a	Acc72EX[i].Data8[16064].a
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
Handshake Bit	9	9	9	9	9	9	9	9	9	9	9	9	9	9
+ Block 3														
| **Channel Type**	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Communication	Undefined	Undefined	Undefined	Communication
| **Size of Channel**	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	15616 bytes	0 bytes	0 bytes	0 bytes	15616 bytes
| **Channel Start Address**	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	Acc72EX[i].Data8[212992].a	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	Acc72EX[i].Data8[212992].a
| **Position of Handshake Cells**	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	IN HANDSHAKE CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	IN HANDSHAKE CHANNEL
| **Size of Handshake Cells**	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	16 BITS	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	16 BITS
| **NetX Handshake Register**	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	Acc72EX[i].Data8[524].a	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	Acc72EX[i].Data8[524].a
| **Host Handshake Register**	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	Acc72EX[i].Data8[526].a	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	Acc72EX[i].Data8[526].a
| **Communication Class**	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	MESSAGING	UNDEFINED	UNDEFINED	UNDEFINED	MESSAGING
| **Protocol Class**	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED
| **Conformance Class**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
| **Number of Subblocks**	0	0	0	0	0	0	0	0	0	9	0	0	0	9
|														
|--- **Subblock 0**										CONTROL				CONTROL
| **Size**										8 bytes				8 bytes
| **Start Offset**										Acc72EX[i].Data8[213000].a				Acc72EX[i].Data8[213000].a
| **Transfer Direction**										OUT (Host System to netX)				OUT (Host System to netX)
| **Transfer Type**										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
| **Handshake Mode**										UNCONTROLLED				UNCONTROLLED
| **Handshake Bit**										0				0
|														
|--- **Subblock 1**										COMMON STATUS				COMMON STATUS
| **Size**										64 bytes				64 bytes
| **Start Offset**										Acc72EX[i].Data8[213008].a				Acc72EX[i].Data8[213008].a
| **Transfer Direction**										IN (netX to Host System)				IN (netX to Host System)
| **Transfer Type**										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
| **Handshake Mode**										UNCONTROLLED				UNCONTROLLED
| **Handshake Bit**										0				0
|														
|--- **Subblock 2**										EXTENDED STATUS				EXTENDED STATUS
| **Size**										432 bytes				432 bytes
| **Start Offset**										Acc72EX[i].Data8[213072].a				Acc72EX[i].Data8[213072].a
| **Transfer Direction**										IN (netX to Host System)				IN (netX to Host System)
| **Transfer Type**										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
| **Handshake Mode**										UNCONTROLLED				UNCONTROLLED
| **Handshake Bit**										0				0
|														
|--- **Subblock 3**										MAILBOX				MAILBOX
| **Size**										1600 bytes				1600 bytes
| **Start Offset**										Acc72EX[i].Data8[213504].a				Acc72EX[i].Data8[213504].a
| **Transfer Direction**										OUT (Host System to netX)				OUT (Host System to netX)
| **Transfer Type**										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
| **Handshake Mode**										BUFFERED, HOST CONTROLLED				BUFFERED, HOST CONTROLLED
| **Handshake Bit**										4				4
|														
|--- **Subblock 4**										MAILBOX				MAILBOX
| **Size**										1600 bytes				1600 bytes
| **Start Offset**										Acc72EX[i].Data8[215104].a				Acc72EX[i].Data8[215104].a
| **Transfer Direction**										IN (netX to Host System)				IN (netX to Host System)
| **Transfer Type**										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
| **Handshake Mode**										UNKNOWN				UNKNOWN
| **Handshake Bit**										5				5
|														
|--- **Subblock 5**										PROCESS DATA IMAGE				PROCESS DATA IMAGE
| **Size**										5760 bytes				5760 bytes
| **Start Offset**										Acc72EX[i].Data8[217088].a				Acc72EX[i].Data8[217088].a
| **Transfer Direction**										OUT (Host System to netX)				OUT (Host System to netX)
| **Transfer Type**										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
| **Handshake Mode**										BUFFERED, HOST CONTROLLED				BUFFERED, HOST CONTROLLED
| **Handshake Bit**										6				6
|														
|--- **Subblock 6**										PROCESS DATA IMAGE				PROCESS DATA IMAGE
| **Size**										5760 bytes				5760 bytes
| **Start Offset**										Acc72EX[i].Data8[222848].a				Acc72EX[i].Data8[222848].a
| **Transfer Direction**										IN (netX to Host System)				IN (netX to Host System)
| **Transfer Type**										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
| **Handshake Mode**										BUFFERED, HOST CONTROLLED				BUFFERED, HOST CONTROLLED
| **Handshake Bit**										7				7
|														
|--- **Subblock 7**										HIGH PRIORITY DATA IMAGE				HIGH PRIORITY DATA IMAGE
| **Size**										64 bytes				64 bytes
| **Start Offset**										Acc72EX[i].Data8[216704].a				Acc72EX[i].Data8[216704].a
| **Transfer Direction**										OUT (Host System to netX)				OUT (Host System to netX)
| **Transfer Type**										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
| **Handshake Mode**										BUFFERED, HOST CONTROLLED				BUFFERED, HOST CONTROLLED
| **Handshake Bit**										8				8
|														
|--- **Subblock 8**										HIGH PRIORITY DATA IMAGE				HIGH PRIORITY DATA IMAGE
Size										64 bytes				64 bytes
Start Offset										Acc72EX[i].Data8[216768].a				Acc72EX[i].Data8[216768].a
Transfer Direction										IN (netX to Host System)				IN (netX to Host System)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode										BUFFERED, HOST CONTROLLED				BUFFERED, HOST CONTROLLED
Handshake Bit										9				9
+ Block 4														
| **Channel Type**	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
| **Size of Channel**	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes

	PROFIBUS-DP Master	PROFIBUS-DP Slave	DeviceNet Master	DeviceNet Slave	CANopen Master	CANopen Slave	CC-Link Slave	EtherCAT Master	EtherCAT Slave	EtherNet/IP Scanner/Master	EtherNet/IP Adapter/Slave	Open Modbus/TCP	PROFINET IO Controller/Master	PROFINET IO Device/Slave
| Channel Start Address	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE
| Position of Handshake Cells	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL
| Size of Handshake Cells	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE
| NetX Handshake Register	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE
| Host Handshake Register	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE
| Communication Class	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED
| Protocol Class	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED
| Conformance Class	0	0	0	0	0	0	0	0	0	0	0	0	0	0
| Number of Subblocks	0	0	0	0	0	0	0	0	0	0	0	0	0	0
|														
+ Block 5														
| Channel Type	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
| Size of Channel	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes
| Channel Start Address	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE
| Position of Handshake Cells	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL	BEGINNING OF CHANNEL
| Size of Handshake Cells	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE
| NetX Handshake Register	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE
| Host Handshake Register	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE
| Communication Class	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED
| Protocol Class	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED
| Conformance Class	0	0	0	0	0	0	0	0	0	0	0	0	0	0
| Number of Subblocks	0	0	0	0	0	0	0	0	0	0	0	0	0	0